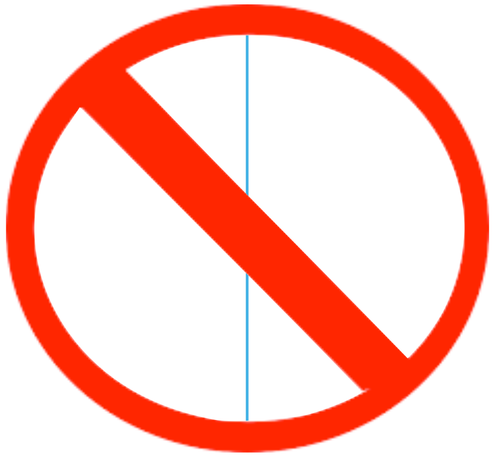
Java + VW

Jon Morra and Ryan Deak

Desired Properties

What we want in a library

Fast



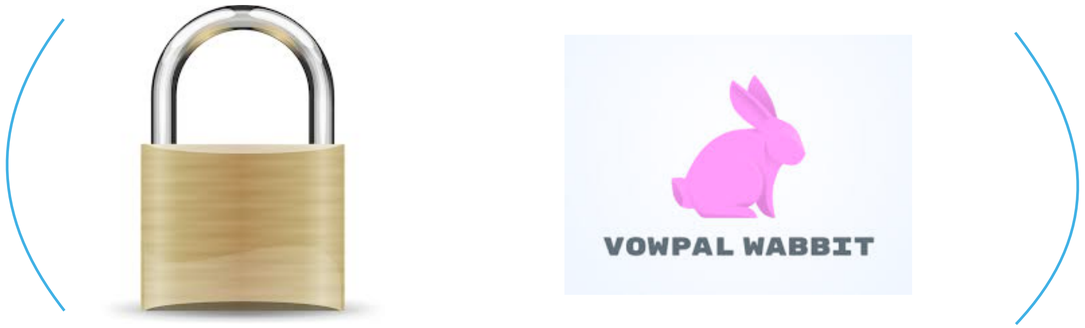
(Process Pipes)

Strong Typing

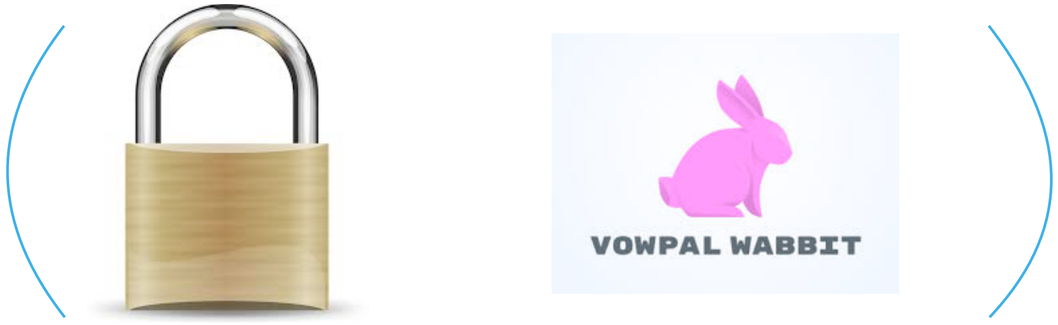
```
public static <T extends VWLearner> T create(final String command) {
    long nativePointer = initialize(command);
    VWReturnType returnType = getReturnType(nativePointer);

    switch (returnType) {
        case ActionProbs: return (T)new VWActionProbsLearner(nativePointer);
        case ActionScores: return (T)new VWActionScoresLearner(nativePointer);
        case Multiclass: return (T)new VWMulticlassLearner(nativePointer);
        case Multilabels: return (T)new VWMultilabelsLearner(nativePointer);
        case Prob: return (T)new VWProbLearner(nativePointer);
        case Scalar: return (T)new VWScalarLearner(nativePointer);
        case Scalars: return (T)new VWScalarsLearner(nativePointer);
        case Unknown:
        default:
            // Doing this will allow for all cases when a C object is made to be closed.
            closeInstance(nativePointer);
            throw new IllegalArgumentException("Unknown VW return type using command: " + command);
    }
}
```

Thread Safety

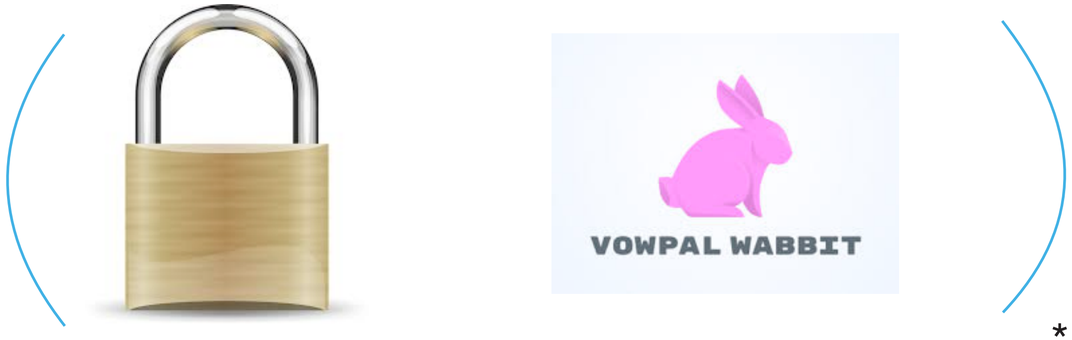
Thread 1



Thread 2



Thread 3



*Except in case of multiple passes

How to build it

```
jon.morra:vowpal_wabbit/ (master) $ make
cd vowpalwabbit; /Applications/Xcode.app/Contents/Developer/usr/bin/make -j 8 things
make[1]: Nothing to be done for `things'.
jon.morra:vowpal_wabbit/ (master) $ make java
cd vowpalwabbit; /Applications/Xcode.app/Contents/Developer/usr/bin/make -j 8 things
make[1]: Nothing to be done for `things'.
cd java; /Applications/Xcode.app/Contents/Developer/usr/bin/make things
<MORE>
jon.morra:vowpal_wabbit/ (master) $ cd java
jon.morra:java/ (master) $ ls -lht target
total 4856
-rwxr-xr-x  1 jon.morra  10000   2.4M Dec  5 17:38 libvw_jni.dylib
jon.morra:java/ (master) $ mvn install
[INFO] Scanning for projects…
<MORE>
Results :

Tests run: 33, Failures: 0, Errors: 0, Skipped: 1
<MORE>
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

https://mvnrepository.com/artifact/com.github.johnlangford/vw-jni

Example Usages

Real Example

```
String cmd = "--quiet --loss_function logistic --link logistic -f foo.model";
VWScalarLearner m = VWLearners.create(cmd);
for (int i=0; i<100; ++i) {
    m.learn( example: "-1 | ");
    m.learn( example: "1 | ");
}
m.close(); // Can be slow if using multiple passes!!!!

float expVwOutput = 0.50419676f;
m = VWLearners.create("--quiet -i foo.model");
assertEquals(expVwOutput, m.predict( example: "| "),  delta: 0.0001);
```

Example Usages

Simple Model

```
VWScalarLearner m = VWLearners.create("--quiet --loss_function logistic --link logistic -f basic.model");
```

float	**learn**(**String** example) Runs learning on example and returns the prediction output.
float	**learn**(**String**[] example) Runs learning on example and returns the prediction output.
float	**predict**(**String** example) Runs prediction on example and returns the prediction output.
float	**predict**(**String**[] example) Runs prediction on example and returns the prediction output.
protected float	**predict**(**String** example, boolean learn, long nativePointer)
protected float	**predictMultiline**(**String**[] example, boolean learn, long nativePointer)

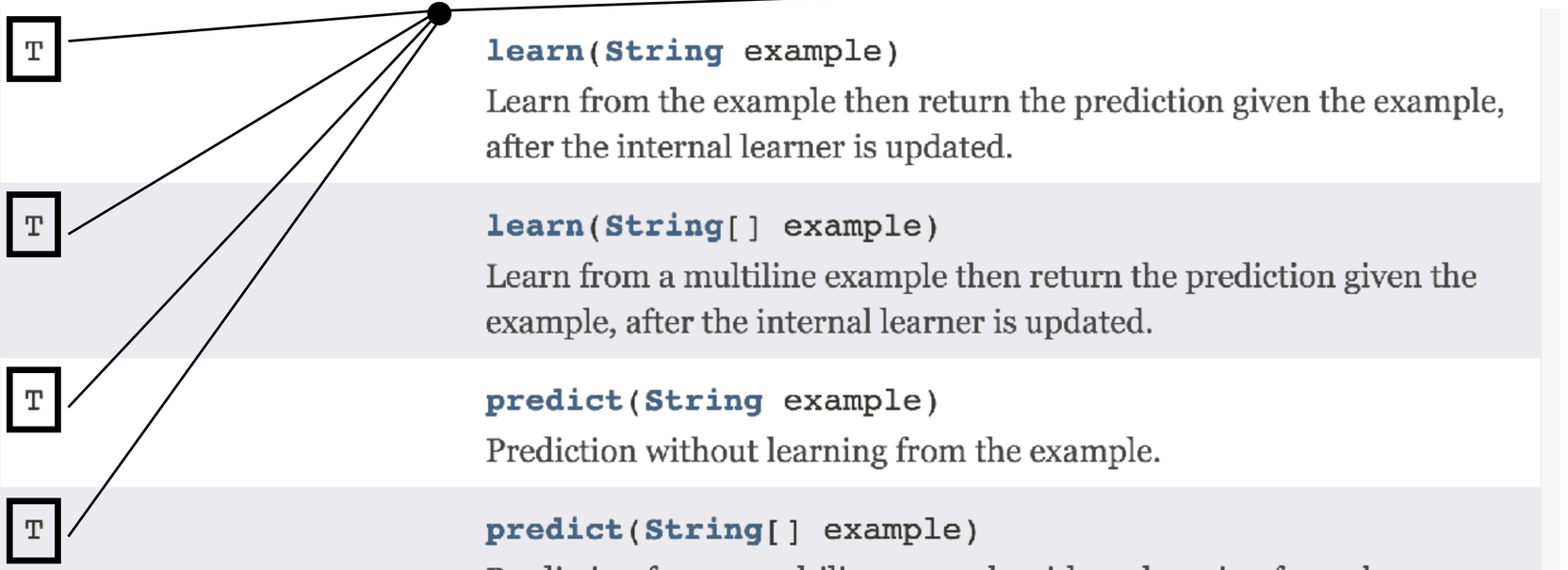
Example Usages

CBADF Model

```
VWActionScoresLearner vw = VWLearners.create("--quiet --cb_adf -f cbadf.model");
```

All Implemented Interfaces:

`Closeable, AutoCloseable, VWLearner, VWTypedLearner<ActionScores>`



T	**learn(String** example) Learn from the example then return the prediction given the example, after the internal learner is updated.
T	**learn(String[]** example) Learn from a multiline example then return the prediction given the example, after the internal learner is updated.
T	**predict(String** example) Prediction without learning from the example.
T	**predict(String[]** example) Prediction from a multiline example without learning from the example.

Future Work

What's next

- More unified build system
- Static linking for JNI layer (only available with Java 8 and above)
- More pure functions in VW that allow easier usage as a library
- Multiple passes requires a global lock

Thank You

Questions?