

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

# Разработка ПО для анализа исходного кода на основе векторного представления AST.

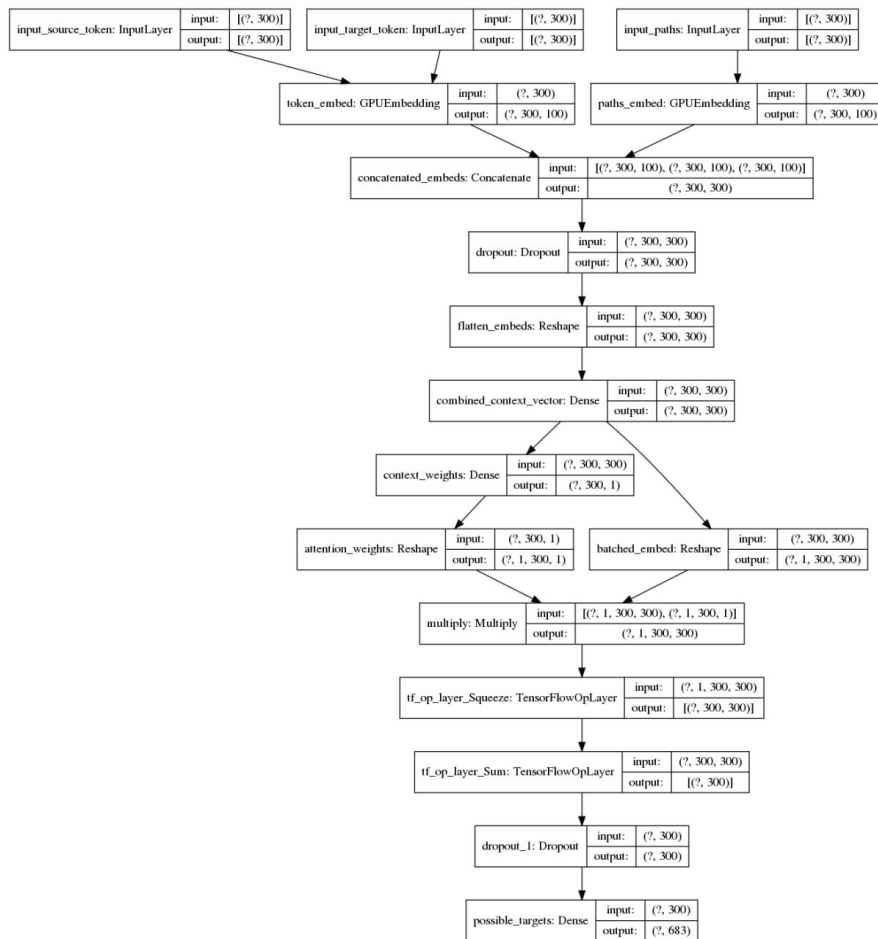
---

Научный руководитель: Бородин А. А.  
Выполнил: Кулагин Р. А.

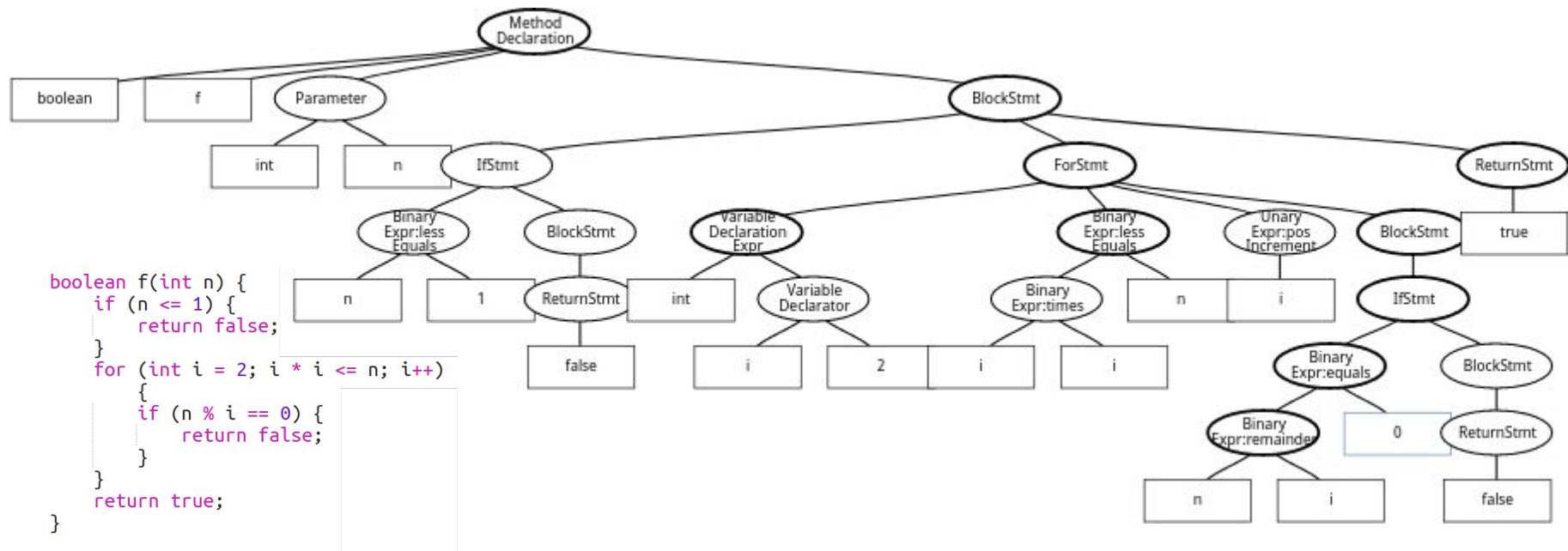
Москва, 2020г.

# code2vec

Эта же архитектура  
легла в основу  
проекта.



# Abstract Syntax Tree



В листьях дерева всегда находятся имена классов, переменных, функций, константы.

# Что делает эта функция?

```
public static void foo(int[] fsbd) {  
    Integer sjkf= fsbd.length;  
    for (int fsak = 0; i < (sjkf - 1); fsak++)  
        for (int fa = 0; < ((sjkf - i) - 1); fa++)  
            if (fsbd[fa] > fsbd[fa + 1]) {  
                int fsck = fsbd[fa];  
                fsbd[fa] = fsbd[fa + 1];  
                fsbd[fa + 1] = fsck;  
            }  
}
```

Пример кода до деобфускации

```
public static void sort(int[] array) {  
    Integer length = array.length;  
    for (int i = 0; i < (length - 1); i++)  
        for (int j = 0; j < ((length - i) - 1); j++)  
            if (array[j] > array[j + 1]) {  
                int value = array[j];  
                array[j] = array[j + 1];  
                array[j + 1] = value;  
            }  
}
```

Пример кода после работы программы (деобфускации)

# code2vec слишком опирается на имена переменных

```
void f() {  
    boolean done = false;  
    while (!done) {  
        if (remaining() <= 0) {  
            done = true;  
        }  
    }  
}
```

## (Correct) Predictions:

done	34.27%
isDone	29.79%
goToNext	12.81%
current	8.93%

```
int f(int n) {  
    if (n == 0) {  
        return 1;  
    } else {  
        return n * f(n-1);  
    }  
}
```

## (Correct) Predictions:

factorial	47.73%
fact	22.99%
fac	9.15%
spaces	7.11%

```
void f() {  
    boolean don = false;  
    while (!don) {  
        if (remaining() <= 0) {  
            don = true;  
        }  
    }  
}
```

## (Incorrect) Predictions:

createMessage	75.07%
checkMessage	16.25%
compareTo	8.55%
putMessage	0.06%

```
int f(int total) {  
    if (total == 0) {  
        return 1;  
    } else {  
        return total * f(total-1);  
    }  
}
```

## (Incorrect) Predictions:

getTotal	84.21%
total	4.06%
average	2.31%
setTotal	2.25%

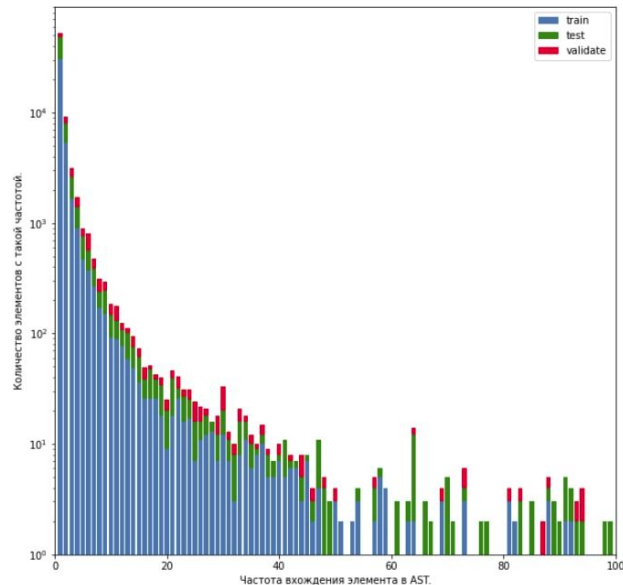
# Нужно обфусцировать!

Случайным образом. Все переменные заменяются на произвольную строку.

(obfuscated-code2vec)

- Много листьев, не несущих в себе никакого смысла.
- Нет связи листьев, используемых для обучения и работы.
- Никак не учитываются имена классов, функций, константы.
- Больше параметров нейросети, обучение идет долго.

Версия с минимальным количеством ограничений, которая у меня запустилась имела более **350 миллионов параметров** (у Сбер ruGPT-3 их всего в 2 раза больше) и практически не работала.



epoch\_loss

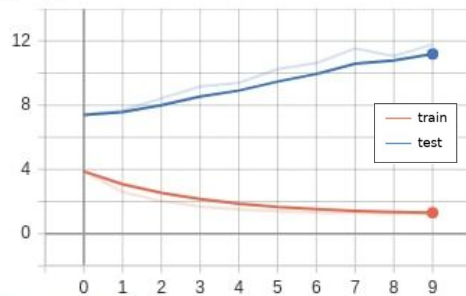
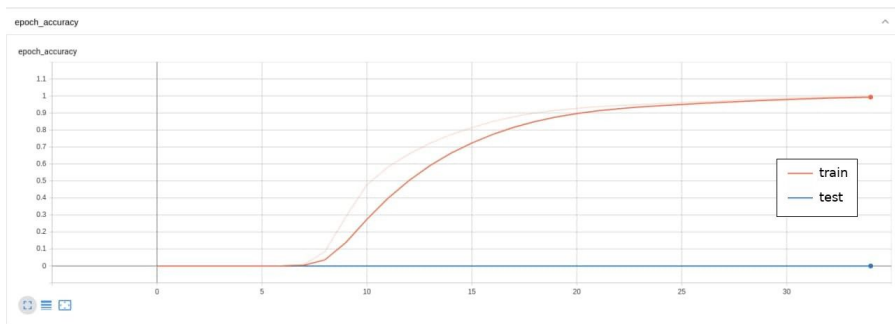


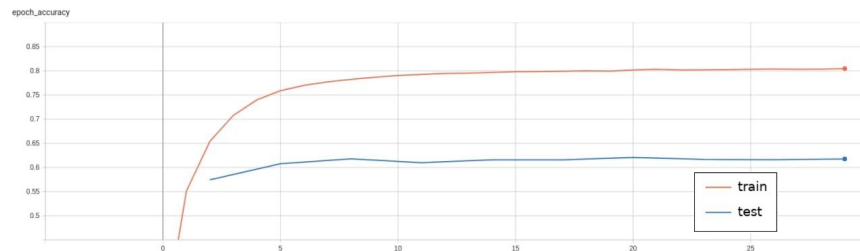
График функции потерь для переменных.  
Точность при обучении 77%

# Единый датасет

Разные проекты, тестовый и обучающий датасета не связаны.



Тестовый датасет – 10% данных, изъятых из обучающего датасета.



# Как оценивать результаты работы нейросети?

Стандартный Accuracy

$$\frac{TP+TN}{TP+FN+FP+TN}$$

Учитываются только точные попадания (ТОП-1) и верно предсказанные непопадания

Самописный Precision

$$\frac{TP}{TP+FP}$$

Попадания в ТОП-5 считаются успешными.

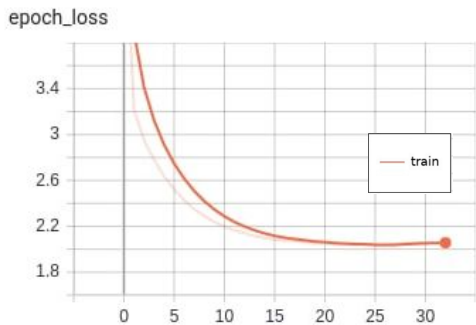


# Меняем обфускацию

Замена, основываясь на типе листа AST.

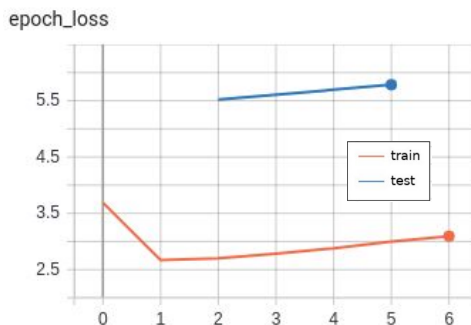
- Мало листьев. Имена классов – CLASS\_NAME, функций – FUNC, константы – CONSTANT.
- Имена встроенных типов и некоторые константы (0, 1, INT\_MAX, NaN, null, "") можно не обфусцировать.
- Одинаковые листья и для обучения, и для работы.

График функции потерь для нейронной сети для определения идентификаторов переменных, обученной на обфусцированных переменных со случайным индексом (в пределах 50).



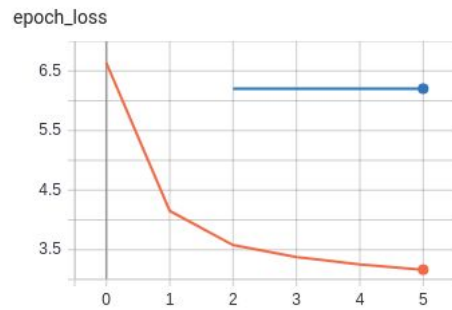
Переменные. Точность: train 15%

График функции потерь для нейронной сети для определения идентификаторов переменных, обученной на обфусцированных переменных без индекса.



Переменные.

Точность: train 46%, test 30%



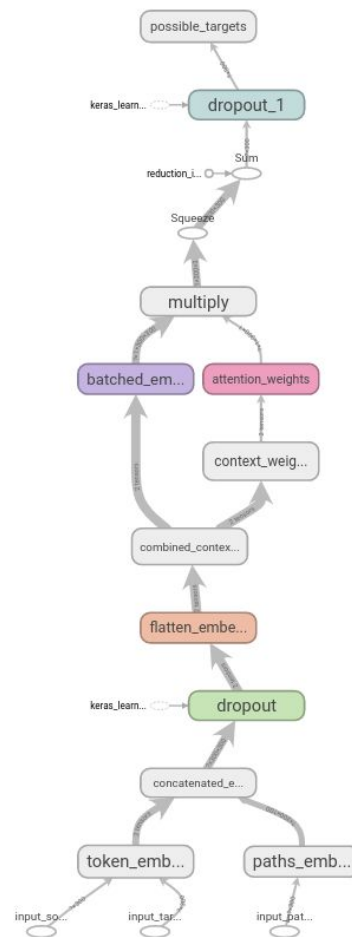
Функции.

Точность: train 46%, test 35%

Демонстрация.

# Особенности архитектуры

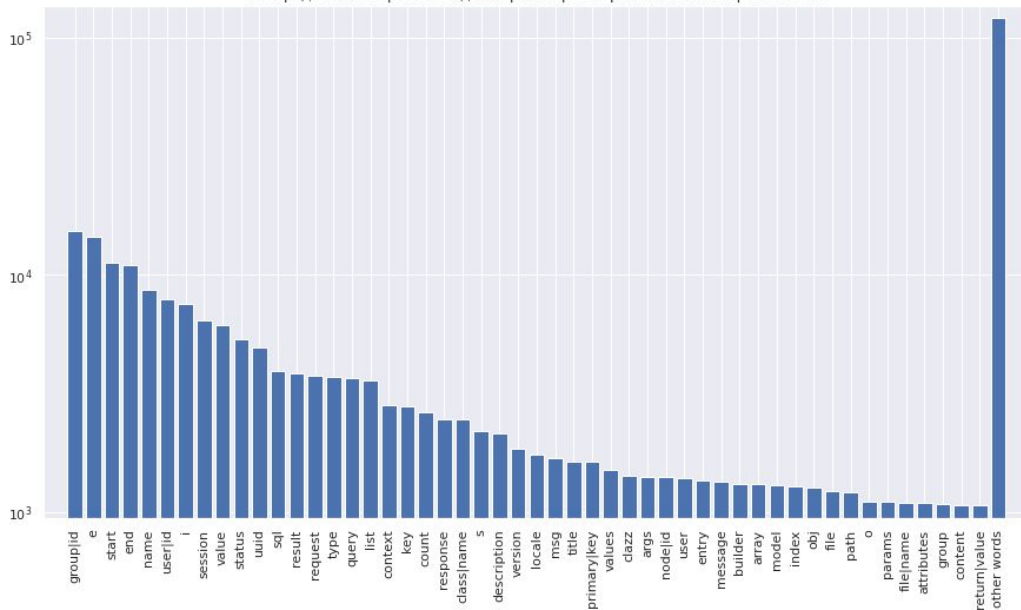
Эмбединги учатся очень быстро, а dense учится долго. В результате мы имеем веса эмбедингов, обучившихся под случайные веса слоев dense.



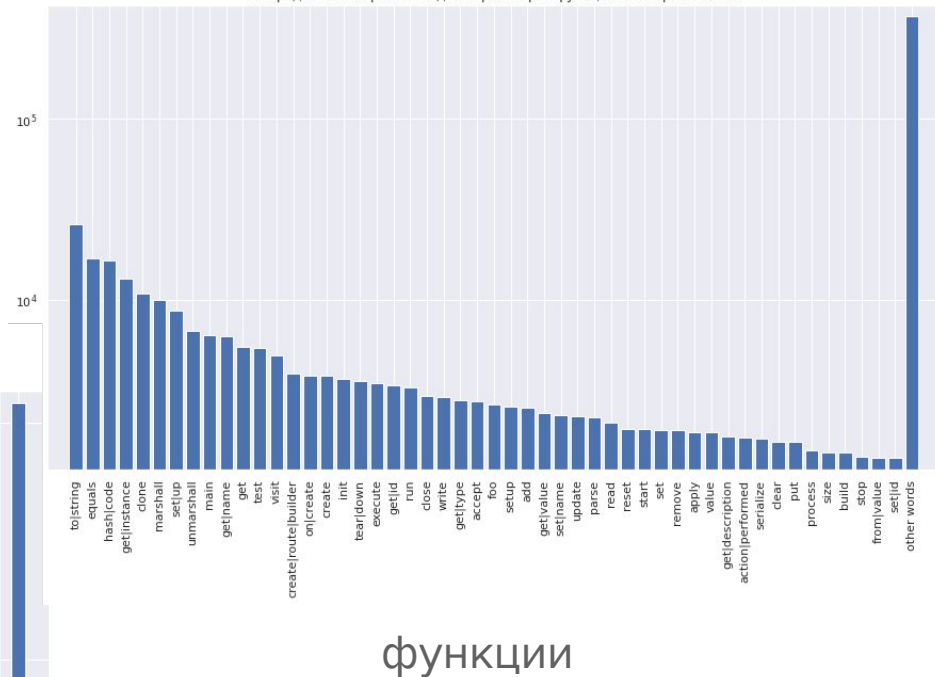
# ТОП-50 идентификаторов

## переменные

Распределение первых 50 идентификаторов переменных по встречаемости



Распределение первых 50 идентификаторов функций по встречаемости



## функции