Bayou

Задача

- Генерация кода
  - Неполная спецификация
  - Вычислительная сложность

Концепция

- Байесовский подход
  - Апостериорное распределение программ по данным evidence
- Скетчи
  - Language agnostic
  - Близки к intent программы
  - Простые
  - Могут быть конкретизированы в программу
- Скрытая переменная intent
  - Неявно моделируем глубинный смысл программы

Workflow

- X - evidence, Y - sketch, Z - intent
  - P(Z|X) -- по evidence получаем распределение intent
  - Сэмплируем Z
  - P(Y|Z) -- по intent получаем распределение sketch
  - Сэмплируем Y
  - Конкретизируем sketch в программу используя P(Prog|Y)

* На каждом этапе может применяться некоторое количество магии (и тервера)

Пример

```
public class Program {
    void foo(String file) {
        Evidence.apicall("readLine");
    }
}
```

```
import java.io.FileReader;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.io.BufferedReader;

public class Program {
    void foo(String file) {
        String s;
        BufferedReader br;
        FileReader fr;
        try {
            fr = new FileReader(file);
            br = new BufferedReader(fr);
            while ((s = br.readLine()) != null) {}
            br.close();
        } catch (FileNotFoundException _e) {
        } catch (IOException _e) {
        }
    }
}
```

Постановка задачи

- X -- evidences, Prog -- program, Env -- type environment
- Задача обучения: Имея корпус *{(X_i , $Prog_i$, Env_i)}* и модель *P(X, Prog | M, Env)* найти

$$\mathbf{M}^* = \arg\max_{\mathbf{M}} \sum_i \log P(\mathrm{X}_i, \mathrm{Prog}_i | \mathbf{M}, \mathrm{Env}_i)$$

- Задача синтеза: Имея evidence X, type env. Env и распределение *P(X, Prog | Env)* получить апостериорное распределение *P(Prog| X, Env, safe(Prog, Env))*

Скетчи

- Y -- sketch
  - Скетчи определены в некоторой грамматике скетчей
  - Она не содержит переменных
  - Она содержит конструкции управления потоком выполнения
- Функция абстракции *a: Prog -> Y*
  - Удаляет из AML* всё, что не присуще грамматике скетчей
- Распределение конкретизации *P(Prog| Y, Env)*
  - Если Y не конкретизируется под Env, то вернётся ⊥

* AML -- процедурное подмножество Java

Реформулировка задачи

- X -- evidences, Prog -- program, Env -- type environment
- Задача обучения: Имея корпус *{(X_i , $Prog_i$, Env_i)}*, *a* и P(X, Y | M) и *Yi = a($Prog_i$)* найти

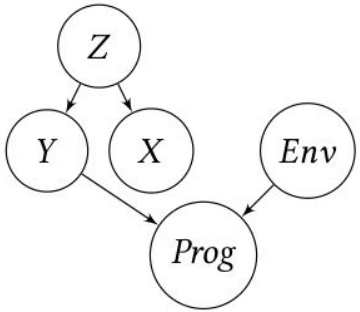
$$M^* = argmax_M \sum_Y logP(X_i, Y_i|M)$$

- Задача синтеза: Имея evidence X, type env. Env и распределение *P(X, Y)* получить апостериорное распределение

$$P(Prog|\mathrm{X}, \mathrm{Env}, safe(Prog, \mathrm{Env})) = \sum_{\mathrm{Y}} P(Prog|\mathrm{Y}, \mathrm{Env}, Prog \neq \perp)\, P(\mathrm{Y}|\mathrm{X}).$$

Intent переменная

- Z -- скрытая переменная выражающая intent программы
  - $P(X_i, Y_i|M) = \int_{\Omega_z} P(X_i, Y_i, Z \mid M)dz$
- P(X, Y, Z) тренируется на корпусе
- В работе используем P(Z|X), P(Y|Z)



BED

- P(Z) -- нормальное многомерное распределение (Normal(O, I))
- P(X|Z) -- encoder
  - Кодируем X_{cxt}, X_{typ}, X_{api} (используем f_{cxt}, f_{typ}, f_{api})
  - P(X|Z) -- сумма произведений нормальных распределений ($\sum_{ev}\prod_{i}\text{Normal}_{ev}(Z, \boldsymbol{\sigma}^2_{ev}I)$)
  - Отсюда выражается и P(Z|X)
- P(Y|Z) -- decoder
  - Используя RNN генерируем скетч
- Тренируется используя variational bound (max L, см. VAE)

$$\log P(\mathrm{X}, \mathrm{Y} \mid \mathbf{M}) = D_{KL}[P(Z \mid \mathbf{M}, \mathrm{X}) \parallel P(Z \mid \mathbf{M}, \mathrm{X}, \mathrm{Y})] + \mathcal{L}$$

- Параметры -- $\boldsymbol{\sigma}_{ev}$, P(Y|Z) params, параметры функций кодирования

Combinatorial concretization

- Сэмплируем программу из P(Prog|Y, Env)
- Но для простых программ можно обойтись random walk
  - Конкретизируем Y до некоторого промежуточного Y’
    - Добавляем пару переменных
  - Находим возможные продолжения конкретизации
  - Случайно выбираем следующее

Общая структура

- Evidence embedding layer
  - LDA преобразующий evidence в распределение над топиками
- Evidence encoder
  - NN преобразующая распределение в d-dim вектор
- Пространство intent переменной
- Intent decoder
  - Две RNN генерирующие скетчи по intent
- Combinatorial concretization
  - Случайный программы конкретизирующей скетч в текущем Env

Эксперименты

- 98619 программ использующих Android API
  - Извлечены evidence
  - Построены скетчи
  - 80% обучение, 20% контроль
- 256, 64 and 32 топика для API calls, types, и contexts соответственно
- 128 узлов в encoder для API calls, 64 для types и contexts
- 256 узлов в decoder
- 32 -- размерность пространства intent
- 100 эпох

	Min	Max	Median	Mean
X_{Calls}	1	9	2	2.6
X_{Types}	1	7	2	1.8
X_{Cxt}	0	11	2	2.1
X	1	23	6	6.5

Топики

	Calls		*Types*		*Cxt*
1.	setJavaScriptEnabled setWebViewClient setWebChromeClient	1.	Builder AlertDialog Dialog	1.	Reader InputStream String
2.	getRunningAppProcesses restartPackage killBackgroundProcesses	2.	InputStreamReader BufferedReader Throwable	2.	Notification Intent Context
3.	setTitle setMessage show	3.	LayoutParams LinearLayout FrameLayout	3.	OnItemClickListener OnScrollListener ListAdapter

Quality scenario

- Запись в файл int
  - API call “write”, type “FileWriter”, context “int “
- InputStream из BluetoothSocket
  - context “BluetoothAdapter”, “String”