1 Немного про грамматики

Предполагаемую структуру свёртки можно описать КС-грамматикой. При этом можно использовать более выразительные конструкции, чем нормальная форма Хомского (НФХ). В результате можно пытаться описать именно вторичную структуру, а не содержимое конкретной цепочки. При этом возможно как уточнение содержимого, так и ослабление структуры.

Конструкции, расширяющие НФХ, доступные для описания вторичной структуры.

- **Регулярные выражения**. Доступны операции $\{+, *, |, ?\}$. Например, любой из символов $\{A, U, G, C\}$, повторённый не менее 1 раза.

```
any : (A | U | G | C)+
```

- **Повторения** позволяют ограничивать количество однотипных элементов (в отличии от $+^*$). Например, повторение `any` от 2 до 5 раз включительно:

```
s : any*[2..5]
```

- **Метаправила** позволяют описывать параметризуемые шаблоны вывода. Например, шаблон вывода списка, пареметризованный элементом списка и разделителем.

```
not_empty_list<item sep>: item (sep item)*
```

  Применение описанного выше шаблона. Фактическими аргументами метаправила могут быть произвольные конструкции: регулярные выражения, другие метаправила.

```
s1: not_empty_list<NUM COLON>
s2: not_empty_list<s1 (DOT|COMMA)>
```

Возможный пример описания свёртки для tRNA.

```
[<Start>]
folded: stem<(any*[1..3]
              stem<any*[4..10]>
              any*[1..3]
              stem<any*[4..7]>
              any*[3..5]
              stem<any*[4..7]>
              )>

stem<s>:
      A stem<s> U
    | U stem<s> A
    | C stem<s> G
    | G stem<s> C
    | G stem<s> U
    | U stem<s> G
    | s

any: A | U | G | C
```

Можно добавить вероятности для фильтрации деревьев. Применимы те же фильтры, что и в работах на основе CYK.

```
[<Start>]
folded: stem<(any*[1..3]
              stem<any*[4..10]>
              any*[1..3]
              stem<any*[4..7]>
              any*[3..5]
              stem<any*[4..7]>
              )>

stem<s>:
      [p1] (A stem<s> U)
    | [p1] (U stem<s> A)
    | [p1] (C stem<s> G)
    | [p1] (G stem<s> C)
    | [p2] (G stem<s> U)
    | [p2] (U stem<s> G)
    | [p3] (s)
```

```
a1: [p1] (A)
  | [p1] (U)
  | [p1] (G)
  | [p1] (C)
```

Грамматика может быть модульной. Это позволяет описать некоторые базовые элементы отдельно и далее использовать, подключая соответствующие модули/модуль. Например, в переиспользуемый модуль можно вынести определение для `stem, any, R, Y` и другие элементы.

После работы алгоритма доступны все деревья разбора, соответствующие хаданной грамматике, а значит и вторичной структуре. Далее, эти деревья сожно обрабатывать отдельно. Например, отбирать более вероятные используя различные фильтры, метрики, шаблоны.

Что можно ещё интересного делать.

- Пытаться восстанавливать грамматики из известных структур свёрток, а не описывать руками [1].
- Восстановливать деревья по известным свёрткам.
- Использовать структурные фильтры для деревьев.

2 Обсуждение

Есть ряд вопросов относительно того, на сколько интересен структурный поиск. С одной стороны, есть работы, так и ли иначе связанные с этим. Например, в работе [3] используют поиск stem-ов и говорят, что это улучшает качество поиска. Работа [4] также говорит о полезности структурного поиска. Есть и другие (например, [2]). При этом есть работы [1, 5] в которых утверждается, что соответствующие грамматики сложны и это проблема. С другой стороны, можно предложить даже более выразительный язык, чем описанный выше (например, добавить возможность указывать диапазон для размера stem-а), что позволит существенно упростить описание шаблонов. Можно также попробовать извлекать их из известных вторичных структур (тот же стокгольмский формат). Интересно ли такое направление?

Был проведён ряд базовых тестов. Для тестов использовались цепочки из тестого набора Infernal и грамматика для tRNA, представленная выше. При первых тестах выявлено следующие.

- Так как поиск исключительно структурный, то находится больше одного вхождения, указанного в тесте. В среднем на 1000 символов 4 непересекающихся участка. Тестировались цепочки до 4000.

При этом находятся, как правило, не целиком цепочки, указанные в базе, а их подцепочки, так как выкидываются несвёрнутые хвосты.

- Есть набор тестов, на которых не находится указанное вхождение. При этом, указанный участок, обработаный инструментом RNAFold действительно сворачивается далеко не в "классический крестик". Значит ли это, что tRNA не описывается структурой? Что это может значить?

- Использование даже простых вероятностных фильтров улучшает результат.

- Производительность и расход памяти пока совсем не гуманны. Но во многом это технические проблемы, с которыми понятно как бороться.

3 Технические детали

Технически, задача достаточно хорошо масштабируемая. Возможно использование смешанных вычислений: CPU + GPGPU. В качестве алгоритма можно применять не только CYK, но и другие алгоритмы, которые могут оказаться более эффективными.

На Infernal я посмотрел. Внутри жуть что происходит. Код хоть и снабжён комментариями, но крайне плохо написан. Я не смог за 2 дня придумать, как быстро его передизайнить, чтобы он работал с графом, хотя для этого требуется просто другая раскладка матрицы, используемой в CYK. Что при аккуратном дизайне не должно вызывать проблем.

Сейчас работаем над демкой структурного поиска и решаем технические проблемы с размерами и представлением даных. А так же исследуем альтернативные алгоритмы синтаксического анализа.

Всё происходящее известным образом обобщается до графов.

Список литературы

[1] Dowell R. D., Eddy S. R. Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction //BMC bioinformatics. — 2004. — Т. 5. — №. 1. — С. 1.

[2] Eddy S. R. A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure //BMC bioinformatics. — 2002. — Т. 3. — №. 1. — С. 1.

[3] Yogev S., Milo N., Ziv-Ukelson M. StemSearch: RNA search tool based on stem identification and indexing //Bioinformatics and Biomedicine (BIBM), 2013 IEEE International Conference on. — IEEE, 2013. — С. 145-152.

[4] Eddy S. R. Homology searches for structural RNAs: from proof of principle to practical use //RNA. — 2015. — Т. 21. — №. 4. — С. 605-607.

[5] Anderson J. W. J. et al. Evolving stochastic context–free grammars for RNA secondary structure prediction //BMC bioinformatics. — 2012. — Т. 13. — №. 1. — С. 78.