Конструирование анаграмм.
Теория и практика.

Марьин О.П.
Москва 2018.

Генерация анаграмм с помощью перебора с использованием анаграммных индексов.

Здесь я опишу простой и практичный метод генерации недлинных (4-5-6-7-8 слов), который используется в программе Комбинаторный Поэт (КП).

Исходный, начально заданный, базовый текст может содержать произвольное количество слов, которые в сумме имеют не очень большую длину (до 12-13 букв). И я предпочитаю именно такие анаграммы. Но есть авторы, сочиняющие 'длинные' анаграммы, в которых начальная часть может состоять из четырёх и более слов, а ее длина доходить до 20 и более букв. Они бывают весьма забавны по смыслу. Точнее, в таких длинных анаграммах легко удаётся воспроизвести любую мысль. К недостаткам длинных анаграмм я отношу то, что их анаграммность воспринимается только головой, умом ('ну анаграмма, и что дальше?'). То есть, ни визуальным, ни слуховым наблюдением она не подтверждается - на слух и вид такие анаграммы уже не верифицируются и (часто, но не всегда) не отличаются от обычного текста. Впрочем, последнее может вызывать уважение читателей, именно потому, что построение достаточно длинного анаграммного текста, с яркой мыслью и правильным синтаксисом кажется им некоей магией. О приёмах генерации 'длинных' анаграмм – следующий раздел статьи. В ней даются некоторые примеры и ‘разоблачается’ магия этого процесса.

В программе КП базовый текст может расщепляться на анаграммный спектр (АС) из 2-3-4-х словных вариантов. Поскольку получение 4-х словных комбинаций занимает больше всего времени, то в КП можно ограничивать генерацию 3-х словными комбинациями. Но по опыту использования программы КП я определил для себя, что в 80% процентов случаев, достаточно 2-х словных вариантов производной (2-й) части анаграммы. Вместе с базовой частью они дают 3-6 словные анаграммы. Что дает вполне качественные анаграммы, которые ещё не теряют ни слуховые, ни визуальные особенности анаграмм.

2-х словные анаграммы строятся за один проход анаграммного мультииндекса. Если не считать начального построения индекса анаграммных ключей (который делается один раз, даже если ищется много анаграмм сразу). При начальном проходе словаря строится мультиндекс с ключами следующего вида:

A1N1.A2N2.A3N3…ALNL

Где A1,A2,A3… - буквы входящие в слово, упорядоченные по алфавиту,

A

N1,N2,N3… - соответствующие числа вхождений этих букв в слово.

Например, для слова 'анаграмма' анаграммным ключом будет строка:
А4Г1М2Н1Р1

Для ключей определены операции сложения и вычитания ключей. Нас больше интересует операция 'вычитания'. Из ключа K1 можно вычесть ключ K2, если они содержат один и тот же набор букв и все количества вхождений букв в ключе K1 больше либо равны соответствующих чисел вхождения букв в ключе K2. Иначе операция запрещается. Результат операции это ключ с таким же или меньшим составом букв и

количеством вхождений, равным разности соответствующих количеств вхождений в ключе K1 и K2. Если разность равна нулю, то буква из ключа исключается.

При таком способе построения ключа все слова-анаграммы будут иметь один и тот же анаграммный ключ, поскольку он точно и однозначно определяет буквенный качественный и количественный состав слова. К одному и тому же ключу будут привязаны все его слова анаграммы (мультииндекс, multimap). И операция перечисления по заданному ключу всех его слов-анаграмм является чрезвычайно быстрой операцией (гораздо более быстрой, чем явный прямой проход по всему списку слов). Обозначим как Anag(K) – множество всех слов, соответствующих ключу K (в сущности его анаграмм – если ключ K превратить в обычное слово).

Для построения двух-словной анаграммы на заданный базовый текст мы строим сначала мультииндекс анаграммных ключей. Затем определяем анаграммный ключ базовой части анаграммы (KBZ). Далее делается ОДИН проход по всем ключам мультииндекса. В цикле на каждом шаге из анаграмного ключа базовой части (KBZ) вычитается текущий обрабатываемый ключ мультииндекса (KT). Если операция не возможна, то переходим к следующему ключу по циклу. Если операция возможна, то мы берём ключ K = KBZ – KT и строим множество (перечисляем) слов из S1 = Anag(K). Также строим множество слов S2 = Anag(KT) и строим (перечисляем) множество - декартово произведение множеств S1 и S2 = S1 x S2. То есть все пары слов (w1,w2) где w1 принадлежит S1, а w2 принадлежит S2. По смыслу данного алгоритма базовый текст, дополненный словами w1 и w2 (любая пара), будет технически представлять собой анаграмму. И w1, w2 дают вторую (производную) часть анаграммы. Обычно для 2-х словных анаграмм получается набор до 300 вариантов, из которого синтаксический и смысловой отбор делается (пока) вручную – что не так уж и сложно при соответствующей сноровке.

Для получения 3-х и 4-х словных комбинаций устраиваются дополнительные циклы по мультииндексу, по аналогичному сценарию.

В программе имеется возможность не только находить 2-х словные (и более) анаграммы, но и искать так называемые дополнения (до анаграмм). Если есть два текста T1 и T2, то можно поставить задачу нахождения слов w1 и w2 таких, что пара (T1 w1) и (T2 w2) будет анаграммой. Эта задача также быстро решается за один проход мультииндекса анаграммных ключей и делается это по алгоритму, похожему на описанный выше алгоритм поиска 2-х словных анаграмм.

Построение длинных анаграмм.

Под длиной анаграммы понимается общее количество букв в анаграмме. Она зависит о длины базового текста и кратности анаграммы (количества строк анаграммы). Поэтому пока будем рассматривать не общую длину анаграммы, а длину базового текста анаграммы.

Какая длина базового текста анаграммы является оптимальной для наиболее полного достижения целей анаграммы как художественного произведения? Простой подсчёт по базе данных существующих анаграмм даёт среднюю длину базового текста в 11.5 букв (функция плотности распределения будет дана в отдельном комментарии). По-видимому это значение и являтся практически обоснованным оптимальным значением.

Значительно меньшие по длине базовые тексты приводят к суперкратким анаграммам, которые часто неполно раскрывают тему (хотя конечно бывают и исключения). Значительно большие по длине анаграммы могут более полно раскрывать тему, но на практике этого часто не происходит.

Неискушенному читателю может показаться, что создание анаграмм с длинным базовым текстом требует особых усилий и/или мастерства. Это справедливо, но лишь отчасти. Существует (относительно) простые способы создавать длинные анаграммы.

1-й способ.

Есть несколько коротких анаграмм:
А1-В1, А2-В2, А3-В3...
Из них легко составить очень длинную анаграмму А1 А2 А 3 ... - В1 В2 В3 ...
Конечно, при этом тексты анаграмм должны иметь смысловую близость.

Пример, есть набор анаграмм.
зелена - не заел,
звенела - не зевал,
дурнела - не удрал,
ерундила - не ударил,
произнесла - не заспорил,
трезвонила - не затворил,
надерзила - не задирал.
удивлена - не удавил,
не изжил, изнежил.

Из него создаётся такая достаточно длинная обычная анаграмма.

1-я строка: звенела, зелена, ерундила, дурнела, трезвонила, произнесла, надерзила, удивлена - не изжил,

2-я строка: не заел, не зевал, не удрал, не ударил, не заспорил, не затворил, не задирал, не удавил, изнежил.

Как видно размер строки этой анаграммы - 65 символов, что почти в 6 раз выше среднего значения.

Аналогичный приём, но в более трудном исполнении, применим, если у вас имеется множество неточных анаграмм, почти-анаграмм. В этом случае ещё приходится им подбирать пару из 'домашних заготовок'.

2-й способ.

Он требует использования программы поиска 'дополнений (см. выше)'. Берём практически любые (но лучше близкие по составу) два слова. Одно из них будет начинать первую строку анаграммы, другое - вторую строку. С помощью программы находим множество слов-дополнений, которые дополняют первую и вторую строку. Если получалась приличная анаграмма, то можно снова выбрать пару подходящих (по смыслу к уже имеющимся) слов и начать аналогичный следующий шаг.

Если хорошей анаграммы не нашлось, то можно попытаться слегка изменить слова дополнения, чтобы смысл в общем тексте появился. Поскольку общей анаграммы в этом случае нет, то необходимо повторить процедуру поиска 'дополнений', считая начальной базой уже полученную почти-анаграмму.

Таким образом можно создавать хорошие по смыслу анаграммы практически неограниченной длины.

3-й способ.

Берем ЛЮБЫЕ два длинных текста (это могут быть просто художественные литературные тексты с очень хорошим смыслом и формой). Скорее всего они рассогласованы.

То есть имеют состав букв:
S A1
S A2
S - это общая часть букв двух текстов. A1 и A2 это лишние (непересекающиеся) множества букв первого и второго текста.

Таким образом, для того, чтобы точно анаграмммировать исходные тексты, необходимо построить анаграммы к множествам A1 и A2. Которые потом приплюсовываются к исходным текстам (противоположным).

Если пересечение (S) значительное, то множества A1 и A2 могут быть сравнительно небольшими. Что облегчит создание их анаграмм. Кроме того, легче будет рассовать (спрятать) слова анаграмм среди текста, состав которого является общим для двух текстов. Чуть позже я проведу экспериментальное обоснование этого метода на примерах из классической литературы.

4-й способ.

Конструирование супердлинных анаграмм. Пока что это идея алгоритма.
Такие вещи имеют чисто теоретический интерес или интерес для тех, кто любит 'мировые' рекорды.

Берём или сочиняем произвольный супердлинный базовый текст. Например, чью-то повесть или роман. Далее начинаем из него ‘вычитать’ другие осмысленные предложения из другого длинного текста (романа) – то есть вычеркиваем буквы из состава базового текста. Делаем так до тех пор, пока какая-нибудь буква не становится по счетчику нулевой в оставшемся базовом тексте. После этого мы можем вычитать только предложения, которые не будут содержать данной буквы. Но таких предложений будет достаточно и мы продолжаем добавлять в анаграмму предложения, не забывая вычитать соответствующие буквы из базового текста. Так делаем пока это относительно легко возможно. Скорее всего после этого останется лишь небольшой кусок, который можно будет проанаграмиравать способами, описанными в других разделах статьи.
В ближайшее время я проведу эксперимент по анаграммированию какого-нибудь толстого романа. В основном чтобы отбить охоту у других к установлению бессмысленных рекордов в этой области.

Частота длин базовых текстов анаграмм.
Первая колонка - длина базового текста.
Вторая - количество анаграмм.

4 => 5
5 => 31
6 => 120

7 => 244
8 => 436
9 => 796
10 => 919
11 => 945
12 => 889
13 => 723
14 => 525
15 => 373
16 => 223
17 => 155
18 => 81
19 => 53
20 => 33
21 => 21
22 => 13
23 => 9
24 => 6
25 => 7
26 => 8
27 => 2
28 => 6
29 => 2
30 => 4
31 => 3
32 => 3
33 => 1
34 => 3
35 => 1
36 => 1
39 => 1
42 => 1

Построение рифмованных анаграмм.

Берем произвольную пару рифмованных слов, например, 'беды' - 'победы'. И используем пункт меню 'Анаграммы дополнения' программы КП, который позволяет добавить к каждому из этих слов по одному слову так, чтобы получилась анаграмма. В данном случае возможны такие варианты:

принимал победы,
припоминал беды!

ждал победы и
поджали беды!

ракета победы
покарает беды!

победы кусались,
беды опускались!

начинка победы -
напичкано – беды!

Поиск нетривиальных анаграмм на примере анаграммы про Капицу.

'Вручную' на слова 'Сергей Капица' приходит в голову сочетание 'по крупицам'. Для уравнивания количества букв и добавления двух букв 'с' и 'е' добавляем ('вручную') слово 'все'.

Имеем заготовку 'Сергей Капица - все по крупицам', которая естественно не является анаграммой.

Теперь играет свою партию программа - она находит 8 'дополнений' к этой паре, которые превращают её в точную анаграмму.

Выбираем одно из них:

Сергей Капица приватному
все по крупицам гарантией.

Добавляем 'вручную' предлог 'с': 'с гарантией'. Эту же букву 'с' добавляем в первую часть: 'с приватному'.

Но 'с приватному' - это бессмыслица. Тут программа играет вторую часть симфонии - ищем анаграммы на сочетание 'с приватному'. И среди 387 двусловных вариантов находится вполне приличное и осмысленное 'справит умно', которое очень удачно дополняет текст и завершает мысль.

После некоей перестановки слов получаем результат:

с гарантией, все по крупицам
справит умно Сергей Капица!

Еще примеры рифмованных анаграмм.

хлеба кусочек и ушка,
и бухло - саке, чекушка!

сеем толково
меткое слово!

дерзай умело,
разумей дело!

Анаграммы - скороговорки.

Анаграммы и скороговорки имеют много общего. Многие анаграммы похожи на скороговорки, особенно кратные анаграммы, состоящие из нескольких строк. Это объясняется повторением одних и тех же букв в различных строках анаграммы. И не

только букв, но и трудных сочетаний букв, что характерно для скороговорок. Ниже приводятся примеры анаграмм-скороговорок.

во лесу лозу вяжу.
увозя улов, слежу.
увезя, улов сложу!

шушера шамкала
шакалам, шушера
шумела - шарашка,
украшаем шалаш!
каша, шум, ералаш!

лохи, они оглохли -
холили хилого, но
он их лихо оголил!

хитрая сорока,
яра исто, кроха.
ораторски хая,
корит хаос рая!

Построение именных анаграмм.

Хорошая анаграмма на фамилию или имя и фамилию человека может стать приятным подарком к его Дню Рождения или к другому празднику. Или подарком просто для хорошего настроения.

Но ее удаётся найти не всегда по вполне объективным причинам - таких анаграмм просто мало. Какую-то анаграмму можно найти всегда, но обычно прямая анаграмма на ФИ редко бывает удачной. И есть приёмы, которые увеличивают вероятность нахождения хорошей анаграммы на ФИ человека.

Для этого надо пробовать все формы, связанные с ФИ человека:
-
ФИ
- разные падежи ФИ

Например, с предлогами

с ФИ
у ФИ.

Можно пытаться использовать уменьшительные варианты имен.

Бывает что имя созвучно фамилии - то есть в них несколько одинаковых букв. В этом случае можно искать анаграмму, где Имя будет в одной строке, а Фамилия в другой строке:

Имя W1
Фамилия W2

где W1 и W2 слова, которые вместе с Фамилией и Именем делают эти строки анаграммой.

Примеры именных анаграмм.

Александр Гумённый,
складен ум гранёный!

лих и в край ломил
Михайлов Кирилл!

лад, сноровка, велик
Александр Воловик!

светел герой
Сергей Летов!

Визульное представление анаграмм.

Каноническое представление анаграмм в виде текста – это формат, когда каждая часть анаграммы располагается на одной отдельной строке. Но для усиления художественного воздействия на читателя части анаграммы могут разбиваться на несколько строк. То есть, с анаграммным текстом допустимы все те виды форматирования, что и с обычным неанаграммным текстом.

Но для отдельных типов анаграмм существуют очень интересные визульные форматы представления. Для миниграмм (определение Федина С.), то есть анаграмм, где базовая часть отличается от производной части расположением одной букв, или анаграмм близких к миниграмм, есть визульный формат, который я назвал ГРАФАНАГРАММЫ.

Графанаграммы - это анаграммы, которые показаны одной строкой (вместо двух) в виде текста из заглавных и строчных букв. Причем строчные (маленькие буквы) могут быть как вверху, так и внизу. Строчные буквы, которые расположены в верхней части строки, относятся к первой строке анаграммы. Строчные буквы, расположенные в нижней части строки, относятся ко второй строке анаграммы. Заглавные буквы относятся к обеим строкам анаграммы: и первой, и второй. Чтение графанаграмм требует небольшого усилия ума и немного похоже на разгадывание ребусов.

Ниже приводятся примеры графанаграмм, которые лучше любых слов объяснят сущность этих визуальных объектов.

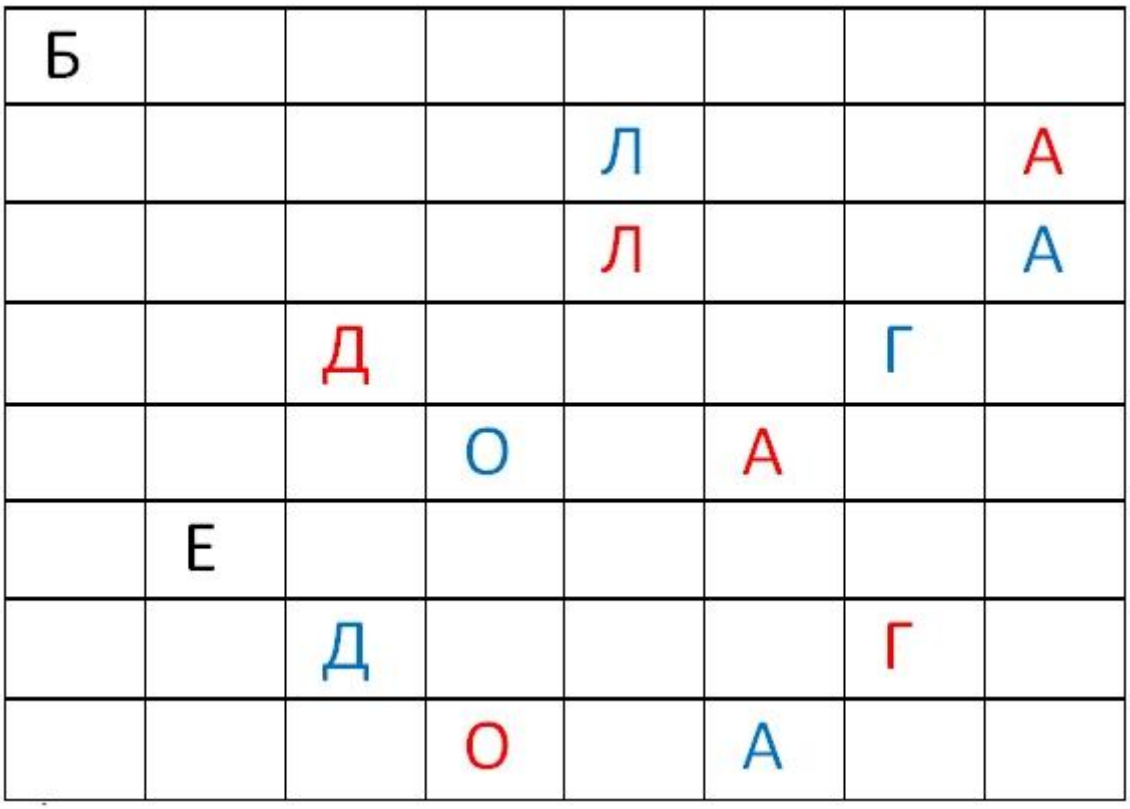
Всего у меня в библиотеке более 250 различных графанаграмм.

Ещё один интересный визуальный формат применим для недлинных анаграмм.

					И	
	С					
				М		
Е						
						Р
		Т				
			Ь			

Здесь первая часть анаграммы читается слева направо, а вторая часть - сверху вниз. Приведённая выше анаграмма записывается так:

есть мир
и смерть!



В этом варианте предыдущего формата первая часть читается слева направо, а вторая и третья строка сверху вниз, но разными цветами. Эта анаграмма записывается как:

бедолага
балда его
благо еда!

Генерация анаграмм с помощью решения задачи целочисленного программирования ЦЛП.

Многие могут подумать, что сведение задачи нахождения анаграмм к задаче какого-то непонятного (или понятного, но, по их мнению, совсем ненужного здесь) целочисленного программирования - это какой-то чисто теоретический трюк, не имеющий особого смысла на практике. В конце статьи я покажу, что это далеко не так, точнее совсем не так. А вначале я просто опишу анонсированный метод.

В анаграммах базовый (фиксированный начальный) текст обычно задан, и к нему необходимо найти вторую часть, вместе с которой они образуют простую (некратную) анаграмму. Для поиска второй части анаграммы важен не сам базовый текст, а состав и количество каждой из входящих в него букв. Буквы можно рассматривать как ресурсы разных типов - всего 33 различных типа ресурсов, то есть 33 различных буквы. А слова в этой постановке рассматриваются как объекты, обладающие определённым набором этих ресурсов. Анаграмма в этом случае будет парой (или даже, в случае кратной анаграммы, не парой, а большим множеством) объектов, обладающих одинаковым набором ресурсов.

Слово, в нашем случае (поиска анаграмм), можно смоделировать целочисленным вектором размерности 33, каждый элемент которого равен количеству вхождений буквы, если в слове есть буква, находящаяся в алфавите на месте, соответствующем слово-вектору (их в слове может быть несколько), и 0 – если её в слове нет. Например, слово 'анаграмма' будет представлено следующим вектором:

[4,0,0,1,0,0,0,0,0,0,0,0,0,2,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

То есть, 4 буквы 'а' – первый элемент = 4, одна буква ‘г’ – четвёртый элемент вектора равен 1, и так далее. Порядок букв в слове для нахождения анаграмм значения не имеет (в отличие, например, от палиндромов).

Если упорядочить множество всех слов в используемом словаре, то любой текст, с точностью до порядка слов, может быть представлен целочисленным вектором большой размерности, где каждый элемент вектора равен количеству вхождений соответствующего слова в текст. Порядок слов в тексте нас конечно интересует и он важен, но определение его мы оставим для другого алгоритма. В практическом плане в нашем случае – коротких текстов анаграмм – с этим легко справится человек (в отличие от задачи нахождения анаграммы).

Можно из векторов слов-наборов ресурсов (подобным вектору слова ‘анаграмма’ приведённому выше, но только в форме вектор-столбца, а не вектор-строки) составить матрицу ресурсов, обозначим её R, поместив эти вектора один за другим в том порядке, в котором мы упорядочили всё множество слов. Она будет размерности [33,N], где N – количество слов в словаре. Поскольку любой текст может быть представлен вектором X размерности N, то умножив эту матрицу на вектор-текст, мы получаем общий вектор ресурсов, размерности 33 (33 буквы), который соответствует суммарному набору ресурсов данного текста. Он равен R*X. В этом векторе каждый элемент даёт общее количество вхождений соответствующей буквы в исследуемом тексте.

Для того, чтобы поставить задачу нахождения анаграммы, мы умножим словарную ресурсную матрицу на вектор-текст D базовой (заданной) части анаграммы. Получим вектор B = R*D.

Будем искать такой вектор-текст X, чтобы R*X = B. Если мы его найдём, это будет анаграмма, соответствующая базовому тексту, из которого был получен вектор 'ресурсов' D. Таким образом, мы получили линейные соотношения, которые сводят задачу к задаче целочисленного линейного программирования.

В задачу ЦЛП обычно ещё входит критерий оптимизации – то есть, линейная функция от искомых переменных (в данном случае от вектор-текста), которую нужно минимизировать или максимизировать. В принципе, для анаграмм это неважно, но рассматривая некоторые грамматические свойства слов, можно поставить и интересные оптимизационные задачи. Например, можно (для заданного базового текста) поставить задачу нахождения анаграммы с максимальным количеством слов, минимальным количеством слов, максимальным количеством существительных или минимальным количество глаголов и так далее.

Например, для нахождения анаграммы с минимальным количеством слов, нужно в качестве критерия оптимизации задать минимум суммы элементов искомого вектор-текста. Сумма элементов вектора это число и его можно получить, умножив вектор-текст на единичный вектор (вектор 'стоимостей’ слов). Это – линейная операция. То есть каждое слово в критерии будет учитываться с одним и тем же весом. Максимум существительных можно получить, задав в качестве критерия произведение вектор-текста на вектор, где каждому существительному будет соответствовать 1, а остальным словам – 0. Можно изобретать различные критерии, меняя этот вектор 'стоимостей' слов, например, где веса существительных, глаголов и др., будут находиться в определённой пропорции.

Но эти же ‘весовые’ суммы-свертки можно задавать не как критерий, а как ограничения. Тогда можно искать анаграммы, которые содержат ровно заданное количество слов, существительных, глаголов или других частей речи. Можно искать анаграммы не с точным значением количеств (слов, частей речи), а количеством, находящимся в заданном интервале – больше 5, меньше 2, и так далее.

Можно задавать ограничения на соотношения между количествами разных частей речи. Например, чтобы число существительных в анаграмме было равно числу глаголов. Или чтобы число прилагательных было меньше или равно число существительных.

Более того, с помощью некоторого приёма можно искать тексты, которые будут соответствовать синтаксически верным предложениям. Например, чтобы найти анаграмму, содержащую текст ПОДЛЕЖАЩЕЕ (ед. ч, ж.р) – СКАЗУЕМОЕ (ед. ч, ж.р) – ДОПОЛНЕНИЕ, нужно приравнять единиц суммы элементов соответствующего типа, которые могут входит в вектор-текст анаграммы:

$$\sum X_{си} = 1 \quad (1)$$
$$\sum X_{г} = 1 \quad (2)$$
$$\sum X_{св} = 1 \quad (3)$$
$$\sum X_{ост} = 0 \quad (4)$$

Где $X_{си}$ – переменные, соответствующие существительным женского рода в единственном числе, именительном падеже , $X_{г}$ – переменные, соответствующие глаголам женского рода в единственном числе, а $X_{св}$ – переменные, соответствующие существительным в винительном падеже, а $X_{ост}$. – все остальные слова. Тогда в решении будет ровно по одному слову указанных типов, и они составят синтаксически верное предложение.

Можно одновременно задать несколько систем соотношений (1), (2) (3) (4), добавив верхние индексы к переменным. Тогда ЦЛП найдёт одновременно несколько синтаксически верных ‘предложений'. Нужно только помнить, что в суммах по ресурсным ограничениям должны участвовать ВСЕ переменные (со всеми индексами).

Можно наоборот, задать соотношения так, что будет найдено только одно из нескольких данных синтаксически верных предложений. Для этого в соотношениях (1), (2) (3) (4) вместо единиц нужно ввести бинарную (0/1) переменную Y (с индексом предложения). И приравнять их сумму единице, так чтобы реализовалось только одно из нескольких возможных ‘предложений’.

Несколько лет назад я реализовал описанную схему для нахождения панграмм (разнобуквиц), которые являются анаграммами алфавита. Словарём были слова разнобуквенные слова, вытащенные из словаря Зализняка. В данном случае вектор в правой части линейных соотношений был единичным – то есть, необходимо было сделать так, чтобы в результирующем тексте было ровно по одной букве алфавита. Соответственно и слов в искомом тексте может быть только по одному и задача ЦЛП превращается в задачу бинарного линейного программирования.

Метод был реализован в программе 'Комбинаторный Поэт'. Решение задачи ЦЛП с 340 тыс. переменных (разнобуквенных слов) и с 35-36 ограничениями (буквы-ресурсы плюс дополнительные ограничения) с помощью пакета LP_SOLVE (бесплатный пакет линейной оптимизации с открытым исходным кодом) занимало от 30 сек и более на ноутбуке Lenovo W510. Чем больше слов запрещено, тем больше времени занимает поиск следующей разнобуквицы. Существуют и более продвинутые пакеты решения задач линейного программирования, которые могут решать задачи с десятками и сотнями миллионов переменных и ограничений. Это CPLEX и GUROBI. CPLEX имеет студенческую, бесплатную лицензию.

При постановке задачи можно было задавать желаемые количества (или интервалы) существительных, глаголов, прилагательных и т д. А также выкидывать

нежелательные слова. Или наоборот задавать вручную начало панграмм. Таким образом, например, я нашёл все свои матерные разнобуквицы. К сожалению, выдаются пока не все решения, а одно. Чтобы получить другое, нужно изменить условия и запустить снова. Изменить условие можно, например, 'запретив' непонравившееся слово. Далее запускаем программу и она находит новую разнобуквицу. В качестве исходного словаря для нахождения разнобуквиц можно взять любой словарь разнобуквенных слов (обычный текстовый файл). В моём начальном словаре было много ‘экзотических’ слов, соответственно – разнобуквиц было много, но много из них было нелитературными, бессмысленными. Чтобы исключить большинство ‘некрасивых’ слов я собираюсь в ближайшее время собрать словарь разнобуквенных слов из слов, входящих в литературные тексты и только них.

Каждый может взять программу Комбинаторный Поэт и попробовать поиграться с разнобуквицами, поскольку это действительно похоже на забавную игру со словами.

Но что же по поводу генерации не разнобуквиц, а простых недлинных анаграмм? Если Вам не нужно находить анаграмму с точно заданной синтаксической структурой результирующей анаграммы, то тогда действительно более практичными будут другие методы нахождения анаграмм, которые я описал в другом разделе статьи. Особенно, если они не очень длинные. Но для нахождения анаграммы с точно и изначально заданной структурой этот метод может оказаться вполне работоспособным.

Кроме того, есть одна задача в которой этот метод может оказаться достаточно эффективным и интересным – это генерация супердлинных анаграмм. Мне самому до рекордов особого дела нет, но некоторые комбинаторики рекорды любят. Поэтому предлагается следующий алгоритм генерации супер длинной анаграммы. Берётся любой заданный текст, очень большой длины – например, повесть или роман целиком. Для первой ЦЛП стадии алгоритма это вообще неважно. Далее мы вычисляем вектор буквенных ресурсов этой повести. И задаём его на вход ЦЛП алгоритма описанного выше. К сожалению ЦЛП выдаст, скорее всего, несбалансированный по частям речи набор слов, из которых трудно будет собрать литературный текст. Можно поставить дополнительное ограничение, на части речи, которое позволит выдать максимально сбалансированный в рамках заданной пропорции набор слов, из которого вручную или с помощью программы (но уже другой) можно будет слепить вполне приличную анаграмму исходной повести.

Другие способы нахождения длинных и супердлинных анаграмм описаны в соответствующем разделе, посвященном длине анаграмм.

Заключение.

Я не 'пишу' анаграммы, 'писать' их невозможно. Это - тексты, написанные не мной... они были там с самого начала. Их возможно только 'вытаскивать' из слов. То, что будет 'вытащено' из слова от меня не зависит практически никак. Я лишь могу отказаться от анаграммы, если она совсем неудачная, или поменять в ней (только) порядок слов в строке или самих строк. Ни одной буквы или слова я ни убрать, ни добавить, ни поменять не могу - то, что получится перестанет быть анаграммой, даже если будет одна лишняя или не та буква.

Анаграммы не являются чем-то исключительным или чем-то особо ценным. Если анаграмма с небольшой основой, то она может выделяться звучанием (так как звучат практически одни и те же буквы в каждой строке) и близка к скороговорке. В большей же степени это словесная игра, прикол. Ценность анаграммы (для специалистов, впрочем и не для специалистов тоже) определяется неформальной комбинацией качеств:

размер основы (одной строки)
кратность (количество строк),

художественный смысл

Размер основы не может быть ни слишком большим, ни слишком маленьким. Большие по основанию анаграммы звучат практически как обычные тексты, вызывая недоуменные вопросы типа 'а в чем тут цимус?' Маленькие по размеру основания анаграммы обычно слишком тривиальны и имеют небольшую кратность. Хотя иногда попадаются забавные экземпляры и среди небольших по размеру основы анаграмм. Кратность - количество строк (повторений) анаграммы. Это важное свойство позволяющее создавать целые мини-произведения. Художественный смысл важен в анаграмме, но может отличаться от обычного смысла. То что 'не имеет смысла' может иметь вполне приличный художественный смысл.