Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Семантическое автодополнение

Курсовая работа студента 445 группы
Василинца Сергея Павловича

Научный руководитель ……. Д.В. Хитров

Санкт-Петербург
2011

Оглавление

1. Введение

1.1 История проблемы

На мобильных устройствах с тачскринами скорость набора текста невелика из-за того, что размеры клавиш малы относительно размеров среднестатистического человека, а также из-за отсутствия тактильного контакта с клавишами. Улучшить ситуацию можно, создавая новые раскладки клавиатуры — на данный момент большинство смартфонов используют стандартную QWERTY раскладку, а также развивая систему автодополнения слов (угадывания следующего слова по уже набранному префиксу). На данный момент системы автодополнения предлагают следующее слово исключительно на основе префикса, не учитывая всего предложения, целью нашего проекта стало исправление этого недостатка.

1.2 Цели работы

Данный проект выполнялся в команде с моим одногруппником, при старте проекта мы не были знакомы с предметной областью, из-за чего на тот момент не могли разделить цели, общими же глобальными целями являлись:

- Исследовать существующие методики семантического автодополнения английского языка.
- На основе исследования реализовать систему автодополнения, основанную на выбранных методиках.

После проведенного исследования моими целями в проекте стали:

- Реализация системы автодополнения на основе статического вывода. (раздел 2.5)
- Анализ и выбор оценочной функции. (Термин будет введен в разделе 2.5)
- Реализация персонифицированной статистики.

2. Обзор

2.1 Обработка естественных языков (Natural Language Processing)

Предметной областью нашей задачи является обработка естественных языков. Задачами, которые исследуются в Natural Language Processing, являются:

- Краткое изложение текста.
- Машинный перевод текста . Эта задача является одной из наиболее сложный в NLP и является одной из задач названных AI-полными, так как требует все возможные типы знаний, которыми обладает человек: грамматика, семантика, факты о реальном мире.
- Разбиение на предложения.
- Распознавание речи.

На основе исследований в Natural Language Processsing, были предложены две техники для решения нашей данной задачи:

- На основе грамматического анализа.
- Статический вывод (Statistical Inference).

2.2 Термины

Введем основные понятия NLP:

N-грамм модели — это модели языка, в которых вероятность следующего слова зависит исключительно от n предыдущих слов.

Корпус — свод знаний о языке в виде списка утверждений «эта n-грамма встречается в текстах с такой частотой»

2.3 Анализ методики на основе грамматического анализа

Первой целью этой методики является построение грамматически верных предложений. Так, например, в английском языке невозможно встретить словосочетания : “must to” или “going watch”. Для того, что избежать таких ошибок надо разобрать структуру предложений, так в данных примерах, выяснить, что это части сказуемого и понять, что они противоречат правилам построения. Но даже правильно построенные с точки зрения грамматики могут быть абсолютно бессмысленны с точки зрения человека, например: «I'm watching the sound».

И тогда второй целью становится категоризация слов, чтобы автодополнение не подсказывало заведомо неправильные варианты. Но исследования показали, что Statistical Inference работает лучше для задачи предсказывания слов[1]. Кратко рассмотрим, почему эта методика неэффективна.

2.3.1 Грамматический разбор предложения

В отличие от многих языков программирования, которые имеют грамматику в математическом смысле, для естественных языков её построить невозможно. Существуют модели, построенные на основе лингвистических исследований, и также существуют статические техники позволяющие делать разбор предложения. Многие из таких техник основаны на Probabilistic Context Free Grammar. Probabilistic Context Free Grammar - это контекстно-свободная грамматика, где каждому правилу сопоставлена его вероятность

2.3.2 Probabilistic Context Free Grammar

Даже в традиционных формах грамматик синтаксис выявляет, как слова группируются друг с другом и относятся друг к другу (как главное и зависящее), и по предложению строят его древовидную структуру.

Частое использование PCFG обусловлено тем, что это наиболее естественная вероятностная модель для древовидных структур. Краткое резюме по техникам основанным на PCFG:

- PCFG хороши для вывода грамматики, так как могут обучаться по исключительно положительным примерам.
- PCFG дают вероятностную модель для английского языка
- На практике эта модель хуже, чем любая n-грамм модель (для $n > 1$), так как n-модели учитывают контекст.

Более эффективные статистические методы для грамматического анализа и вывода грамматики основаны на обучении модели на уже правильно разобранных примерах предложений. Банк таких предложений существует, это — Penn Treebank[5]. Но ввиду того, что этот банк давно не развивается (с 1999), тяжеловесности методов для вывода грамматики и исследований, указывающих на неэффективность данных методов для нашей задачи[1], было решено отказаться от дальнейшего исследования этого пути развития.

2.4 Статистический вывод (Statistical Inference)

Статистический вывод (англ. statistical inference) — использование выборочной информации для получения некоторого представления о свойствах генеральной совокупности.[6]

В общих чертах словах этот метод основан на крайне простой идее. Сначала на наборе текстов собирается частотная статистика об использовании n-грамм. Далее, от данного на вход предложения и префикса мы оставим только последнюю (n-1)-грамму и будем считать вероятности все возможных n-грамм, начинающихся с полученной (n-1)-грамму и заканчивающихся на слова с данным префиксом. Для многих такой подход, откидывающий большую часть предложения и ни как не учитывающий структуру предложений, покажется абсурдным. Но в реальности семантика, базовые синтаксические отношения, проявляющиеся в таком локальном контексте являются хорошими предсказателями, и такие системы довольно хорошо работают.[2]

Данный метод разделяется на две подзадачи:

- Сбор статистики.
- Выбор оценочной функции.

2.5 Оценочные функции (Estimators)

Оценочная функция — функция, которая для n-граммы выдает её вероятность, то есть $P(w_1...w_n) = a$

Пусть мы тренируем нашу модель на тексте из N слов. Можно считать, что наш корпус суммарно содержит N n-gram. Тогда B — количество классов эквивалентности, и оно равно V^n , где V — размер словаря, то есть количество различных слов в тексте.

Назовем $C(w_1...w_n)$ — количество вхождений n-граммы $w_1...w_n$ в данный текст.

$N_r = |\{w1...wn: C(w_1...w_n) = r\}|$ — количество n-грамм, которые вошли в текст ровно r раз.

Оценочные функции бывают двух видов статистические и смешивающие.

2.5.1 Статистические оценочные функции (Statistical Estimators)

Статические оценочные функции — функции, которые выдают вероятность n-граммы, основываясь только на данных для фиксированного n.

Теперь перейдем к рассмотрению возможных функций.

2.5.2 Maximum Likelihood Estimation

$P(w_1...w_n) = C(w_1...w_n) / N$

Эта функция не подходит для статического вывода в NLP. Проблема в том, что наша информация разрежена, даже при использовании большого корпуса. В то время, как небольшая часть слов очень часто встречаются, огромное же число слов крайне редко, и n-граммы, содержащие их, тем реже встречаются, чем длиннее n-грамма. MLE функция присваивает нулевую вероятность всем не встретившимся событиям.

Как пример разреженности информации: после сбора статистики на 1.5 миллионе слов из IBM Laser Patent Text corpus при тестировании на остальных текстах из этого корпуса, только 23% триграмм были ранее встречены. Этот корпус крайне мал по современным стандартам, и кто-то может надеяться, что, собрав гораздо большую статистику, разреженность уйдет. На практике это никогда не будет решением проблемы. Пока есть ограниченное число частых событий в языке, всегда существует почти бесконечный «хвост» бесконечно редких событий. На практике используют другие оценочные функции, и наиболее используемая и дающая лучшие результаты — Good Turing Estimator.[1]

2.5.3 Good-Turing оценочная функция (Good -Turing estimator)

Эта функция оценивает вероятности событий на предположении, что их распределение биномиальное. Этот метод хорош для большого количества наблюдений и хорошо работает для n-gram за исключением того факта, что n-граммы не распределены биномиально.

Функция выглядит:

При $C(w_1...w_n) = r > 0$, $P(w_1...w_n) = r'/N$, где $r' = (r + 1) * S(r + 1) / S(r)$,

иначе, $P(w_1...w_n) = \dfrac{1 - \sum_{r=1}^{\infty} N_r \dfrac{r'}{N}}{N_0}$,

где S(r) — выравнивающая функция для $E(N_r)$, где E – мат. ожидание.

В нашей работе мы использовали $S(r) = a * r^{-1}$, такая функция была выбрана на основе исследований представленных в публикации “Good–Turing Frequency Estimation Without Tears”[4].

2.5.4 Смешивающие оценивающие функции (Combining Estimators)

Для того, чтобы уменьшить разреженность данных, можно комбинировать данные из разных моделей, то есть, например, для подсчёта вероятности использовать одновременно триграммы, биграммы и униграммы.

2.5.5 Линейная интерполяция

Первым примером смешивающих функций является линейная интерполяция, которая выглядит следующий образом:

$P(w_i \mid w_{i-1}...w_n) = a_1P_1(w_i) + a_2P_2(w_i \mid w_{i-1}) + \ldots + a_{n-1}P_{n-1}(w_i \mid w_{i-1}...w_{n-1})$, где $0<=a_i<=1$ и $\sum_i a_i = 1$, где P_i - статистическая функция для i-грамм.

2.5.6 Katz's Backing-off Model

Второй часто используемой функцией является Katz's Backing-off Model, которая выглядит:

При $C(w_1...w_n) > k$:

$$P(w_i \mid w_{i-n+1}...w_{i-1}) = (1-d_{w_{i-n+1}...w_{i-1}})\frac{c(w_{i-n+1}...w_i)}{c(w_{i-n+1}...w_{i-1})},$$

Иначе:

$$P(w_i \mid w_{i-n+1}...w_{i-1}) = a_{w_{i-n+1}...w_{i-1}} P(w_i \mid w_{i-n+2}...w_n)$$

Идея, использованная для данной модели, крайне проста. Если такая n-грамма встречалась — посчитаем её вероятность, если же нет — первое слово отбрасывается, и проверяется (n-1)-грамма и так далее. [3]

2.5.7 Выбор оценочных функций

Мы использовали корпус, состоящий из униграмм, биграмм и триграмм. Было проведено тестирование: брались два первых слова из триграммы и одна буква последнего, и считался процент угадывания третьего слова нашей системой. На основе данных тестов были выбраны Good Turing функция, как статическая функция, и линейная интерполяция, как смешивающая.

2.5.8 Линейная интерполяция vs Katz's Backing-off Model

В результате описанного теста при Katz's Backing-off Model функции проценты угадывания слова были ниже в среднем на 10%. Для того, чтобы объяснить эту разницу рассмотрим следующий пример. Пусть нам даны триграммы A B C и A B D соответственно с вероятностями d и f, где d > f (но сопоставимы, $d/f \approx 1$), и пусть биграммы B C и B D имеют вероятности e и g, причем g >> e ($e/g \approx 0$). В таком случае Katz's Backing Off функция выберет как продолжение C, несмотря на то, что биграмма B D гораздо более популярнее.

Что выберет функция линейной интерполяции, будет зависеть от коэффициентов при статистических функциях.

3. Реализация

В итоге была реализована система автодополнения, основанная на статистическом выводе и использующая статическую Good-Turing функцию и смешивающую функцию линейной интерполяции.

Ядром системы являются классы Vocabulary и NGramModel.

Vocabulary — класс, отвечающий за загрузку словаря из файлов с диска и индексацию слов.

NGramModel — сущность, соответствующая n-грамм модели, то есть она хранит частотную статистику n-грамм для фиксированного n, а также умеет её строить по данному файлу и словарю.

Также реализованы различные статистические (MLE функия. Good-Turing функция) и смешивающие оценочные (Katz's Backing-off, линейной интерполяции) функции, работающие на статистических данных, полученных из NGramModel.

В виду того, что система должна максимально быстро обрабатывать запросы вида список слов и префикс следующего, было решено отказаться от SQL решений из-за их тяжеловесности, в пользу более легковесного и максимально-оптимизированного как по затратам по памяти, так и по скорости решения, реализованного нами самими. Так, класс Vocabulary является неизменяемым (immutable) после того, как единожды был вызван метод build(), что позволило реализовать поиск индексов слов, начинающихся на данный префикс, за O(log(n)), где n - размер словаря, при помощи бинарного поиска, и реализовать поиск слова по индексу за O(1).

Класс NGramModel работает хранит все n-граммы по индексами, полученными из IVocabulary (интерфейс, который реализует Vocabulary, и в котором находятся все основные методы: поиск индексов по префиксу, поиск по индекса по слову), который передается модели при построении.

3.1 Оптимизация использованная в NGramModel

Так как цель нашей системы — угадать следующее слово по его префиксу, в NGramModel приходят n-граммы, по которым она должна вернуть их частоту. При этом мы перебираем все слова начинающиеся на данный префикс, и множество запросов содержат n-граммы одинаковые за исключением последнего слова. Поэтому был сделан следующий цикл обработки (workflow) запросов для NGramModel:

- При начале обработке запроса (даны предыдущие слова и префикс нового).

вызывается метод startPredict, принимающий на вход список слов history. Этот метод находит все n-граммы начинающиеся со слов, находящихся в history.

• Дальше, перебирая слова имеющих данный префикс, вызывается метод frequency с параметром id, который возвращает частоту n-граммы, начинающейся с List<String> history, с которым был вызван до этого startPredict, и заканчивающейся на слово с заданным id.

• Дальше вызывается метод stopPredict, который заканчивает процесс обработки и удаляет ссылки на ненужные более ресурсы, позволяя тем самым их убрать сборщику мусора (Garbage Collectore).

Из-за того, что запросы могут приходить одновременно из разных потоков, эти операции являются локальным для каждого потока (ThreadLocal).

4. Персонифицированная статистика

Для повышения точности автодополнения было решено собирать и использовать статистику использования слов и словосочетаний конкретным пользователем. Причинами создания персонифицированной статистики были следующие тезисы:

- Человек в своей речи использует очень ограниченный запас слов и словосочетаний, и он гораздо меньше, чем наш основной корпус.
- В наше корпусе могут отсутствовать некоторые слова, которые использует человек, так как эти слова недавно родились или являются профессиональным сленгом.

4.1 Реализация

Главным отличием персонифицированной статистики от той, что лежит в ядре нашей системы, является то, что она может менять в реальном времени. Поэтому реализация Vocabulary, которая является неизменяемой (immutable) после построения, не подходит. Поэтому был создан класс EditableVoc, который позволяет добавлять новые слова на лету, но из-за этого более требователен по памяти и по скорости, но что несильно влияет на итоговую скорость обработки запросов ввиду того, что он хранит исключительно слова специфичные для пользователя.

5. Тестирование системы

Здесь представлены краткие результаты тестирования (полную статистику и условия экспериментов можно найти в работе Удалова А. "Семантическое автодополнение", 2011).

Было проведено несколько экспериментов над системой, которая использует корпуса из 1-грамм, 2-грамм и 3-грамм с оценочной функцией Гуда-Тьюринга и некоторыми разумными коэффициентами (далее в таблицах ”1g,2g,3g+users”), в противовес «несемантической» системе (”1g”) – которая использует только корпус из 1-грамм (иными словами, это первоначальный подход, который сортирует слова по убыванию частоты встречаемости в языке и возвращает несколько первых, начинающихся на данный префикс). Также была рассмотрена система, запущенная без пользовательского режима (”1g,2g,3g”).

Результаты экспериментов представлены в таблице ниже. В столбцах указаны два числа через запятую – первое значит длину префикса, предоставленного системе, второе – минимальную длину слова, для которого этот тест запускался. Например, последний тест ”3,8” означает, что системе предлагалось дополнить все возможные n-граммы из предложений исходного текста, такие, что в следующем слове как минимум 8 букв, и при этом ей давался префикс из 3 букв

В таблице показаны частоты попадания правильного слова на одно из первых трёх мест в подсказке (тест возник из первоначальной цели курсовой и он обусловлен опытом набора текста на мобильном устройстве, при котором было бы вполне удобно видеть в качестве подсказки правильное слово на первых трёх местах).

алгоритм	**3,4**	**2,3**	**1,2**	**3,8**
1g	61.70%	52.79%	42.36%	22.63%
1g,2g,3g	73.20%	64.61%	55.88%	42.76%
1g,2g,3g+users	73.58%	64.92%	56.29%	44.08%

6. Результаты проекта

Так как проект выполнялся в команде с моим одногруппником, результаты разделены на две части – результаты всего проекта и результаты моей работы.

В результате всего проекта были реализованы:

- Инструмент для обработки статистики n-грамм с GoogleLabs.
- Система автодополнения, основанная на 3-грамм модели.
- Система для сбора и использования пользовательской статистики.
- Инструмент для сбора статистики из социальной сети Twitter.
- Веб-интерфейс для нашей системы.

Части проекта и исследования, проведенные мной:

- Исследование грамматической методики для решения данной задачи.
- Исследование и выбор оценочных функций.
- Участие в реализации и оптимизации ядра системы автодополнения.
- Реализации персонифицированной системы автодополнения.

По результатам статистики тестирования, указанной в 5-ом разделе, видно что семантическая система автодополнения работает заметно лучше. Тот факт, что она работает не медленнее несемантической, оправдывает её применение в реальности.

Развитие проекта мы видим в следующих направлениях:

- Добавление различные источники для статистики слов (форумы, чаты)
- Уточнение констант. В итоге работы возникло много констант(например, коэффициенты в функции линейной интерполяции) подобранных эмпирическим путём. Уточнение констант можно произвести при помощи Expectation Maximization (EM) algorithm или при помощи нейронных сетей.
- Использование 4-грамм. На данный момент система использует только 3-граммы, ввиду того, что для 4 - грамм нужна статистика очень больших размеров, чтобы данные не были слишком разреженными.
- Исследование различных источников контекста для автодополнения. (Выделение общей тематики сообщения, выделение категорий пользователей по их стилю общения и использование общей статистики для них).
- Создание Android приложения на основе созданного веб-сервиса. Изначально идея родилась в контексте мобильных устройств.

7. Список Литературы

[1] Christopher D. Manning and Hinrich Schütze. 1999. Foundations of Statistical Natural Language Processing. Cambridge, MA: MIT Press.

[2] Collins, Michael John. 1996. A new statistical parser based on bigram lexical dependencies.

[3] Collins, Michael John, and James Brooks. 1995. Prepositional phrase attachment through a backed-off model

[4] William A. Gale and Geoffrey Sampson. Good–Turing Frequency Estimation Without Tears. 217–37 of the Journal of Quantitative Linguistics, vol. 2, 1995

[5] The Penn Treebank Project. http://www.cis.upenn.edu/~treebank/

[6] Wikipedia. Статический вывод. http://ru.wikipedia.org/wiki/Статистический_вывод.