Глава 1

Page Rank

1.1 Введение

Имеется некоторый фрагмент сети интернет, полученный, напрмер, в результате запроса пользователя. Необходимо определить, какие страницы содержат наиболее подходяющую и достоверную информацию. Иными словами, необходимо определить некоторый коэффициент, по которому можно сравнивать страницы между собой. Одним из возможных решений является PageRank.

1.2 Описание

PageRank - общее название группы алгоритмов ранжирования страниц, использующих в основе вычисление параметра, называемого рангом страницы. Ранг страницы - числовой коэффициент, зависящий от количества входящих ссылок и значимости страниц, с которых они исходят. Алгоритмы работают с структурой ссылок страниц интернета, представленными в виде ориентированного графа, в котором вершины - страницы (документы), а ребра - существующие ссылки на внешние страницы.

Существуют различные подходы к подсчету ранга страниц, отличающиеся затраченным временем, точностью и представлением вектора рангов. Некоторые из них будут описаны в следующей главе.

Глава 2

Алгоритмы

2.1 Классы данных

TransitionMatrix Класс представляет собой обертку над матрицей. В данной матрице M:

$M[i][j] = 0$, если не существует прямой ссылки со страницы i на страницу j,

$M[i][j] = 1$, если ссылка со страницы i на страницу j существует.

TransitionProbabilityMatrix Класс представляет собой обертку над матрицей. В данной матрице M:

$M[i][i] = d$,

$M[i][j] = p_{i,j} * (1 - d), i \neq j$, где d - коэффициент затухания или

вероятность пользователя остаться на странице, 1 - d - вероятность перейти на другую страницу, $p_{i,j}$ - вероятность перехода с i страницы на j.

TransitionList Класс представляет собой набор списков ссылок для каждой страницы, то есть:

Для i-го списка наличие элемента со значением j означает наличие ссылки с i страницы в j.

2.2 Интерфейс алгоритмов

Запуск Все алгоритмы реализованы классами и описаны в файле ComputingMethod.py. Все объекты инициализируются некторыми стартовыми параметрами, после чего их можно использовать следующими путями:

1) $iterating\_run(N)$ - запуск N итераций выбранного алгоритма

2) $stopping\_run(\epsilon)$ - запуск алгоритма до срабатывания критерия остановки, где ϵ - заданная точность вычислений

Замечания Не все классы определяют сразу оба пути использования алгоритмов, некоторые из них содержат только один из них. При использовании нереализованного функционала происходит оста-

новка программы с $Exception("Noavailableimplementation")$.

Все классы опираются на метод $\_iteration$, осуществляющий одну итерацию алгоритма. Его использование возможно, но приведет к потере функционала либо производительности данного метода.

Все алгоритмы поддерживают изменение коэффициента затухания d теми или иными способами, но также имеют значение по умолчанию $d = 0.85$, так что задавать его вручную не обязательно.

Все итеративные алгоритмы по умолчанию определяют некоторое значение точности ϵ, все представленные тесты были запущены с базовыми значениями параметров.

Данные Классы содержат три ключевых поля:

$\_iterating\_run\_time$ - время выполнения последнего $iterating\_run$, None по умолчанию

$\_stopping\_run\_time$ - время выполнения последнего $stopping\_run$, None по умолчанию

$\_stat\_vector$ - вычисленный вектор, формат вектора определяется алгоритмом, как и его значение по умолчанию

2.3 Итеративные алгоритмы

Цепь Маркова

Описание Имеется матрица переходных вероятностей Марковской цепи A, где $A[i][j], i \neq j$ - вероятность перехода со страницы i на страницу j. $A[i][i] = 0$. d - коэффициент затухания или вероятность пользователя остаться на странице. 1 - d есть вероятность перейти на другую страницу. Матрица H есть преобразованная следующим образом матрица A:

$H[i][i] = d$,

$H[i][j] = A[i][j] * (1 - d), i \neq j$

Данный алгоритм итеративно ищет приближенное значение вектора статистических вероятностей Марковской цепи π, то есть на каждой итерации k вектор статистических вероятностей равен:

$\pi_k = \pi_{k-1} * H$, при этом $\pi_0 = (1/N, 1/N, ...1/N)$, где N - количество страниц.

Критерием остановки является достаточная близость статистических векторов на текущей и предыдущей итерации или:

$|\pi_k - \pi_{k-1}| < \epsilon$, где ϵ - заранее заданная точность вычислений.

Реализация Алгоритм реализован классом MarkovChain. При создании объекта класса необходимо передать TransitionProbabilityMatrix. Необязательным параметром является стартовый статистический вектор, который по умолчанию задается как вектор одинаковых вероятностей $1/N$, где N - количество страниц.

Реализация поддерживает оба формата запуска.

Сходимость] Вектор рангов на итерации k представляет из себя вероятности оказаться в каждой вершине на шаге k при условии, что начальная страница выбирается случайно с равными для всех страниц вероятностями. Данный вектор сходится к вектору предельных вероятностей.

Power Method

Описание Имеется матрица переходов A, где:

$[i][j] = 0$, если не существует прямой ссылки со страницы i на страницу j,

$[i][j] = 1$, если ссылка со страницы i на страницу j существует.

Каждая страница имеет некоторый вес, при этом:

1) Собственный вес страницы W есть сумма всех входящих весов ссылающихся страниц с коэффициентом d плюс константа $1 - d$: $W = d * \sum W_{in} + 1 - d$

2) Исходящий вес страницы есть W/M, где W - собственный вес страницы, M - количество ссылок.

Алгоритм итеративно ищет приближенное значение вектора весов страниц: на итерации k выполняется пересчет ранга для каждой

страницы.

$r_j = W_j = d * \sum W_{in} + 1 - d$

Критерием остановки является достаточная близость векторов весов на текущей и предыдущей итерации или:

$|\pi_k - \pi_{k-1}| < \epsilon$, где ϵ - заранее заданная точность вычислений. $\pi_k = (r_1^k, r_2^k, ..., r_N^k)$, где r_j^k - ранг j страницы на шаге k.

Реализация Алгоритм реализован классом PowerMethod. При создании объекта класса необходимо передать TransitionMatrix. Необязательным параметром является стартовый вектор весов, который по умолчанию задается как вектор едениц. Коэффициент затухания также может быть передан.

Реализация поддерживает оба формата запуска.

Сходимость] Вектор рангов может быть однозначно опрделелён в условиях, когда в графе отсутствуют циклы. Если же граф содержит цикл, то при увеличении веса i вершины цикла на s:

$W_{newi} = W_{oldi} + s$

$W_{new(i+1)} = W_{old(i+1)} + d * s$

$W_{new(i+2)} = W_{old(i+2)} + d^2 * s$

...

$W_{new(i+k)} = W_{old(i+k)} + d^(k) * s$

...

Так как $d < 1$, то последовательность дополнительный сумм $d^(n) * s$ будет сходиться к нулю при любом конечном s. Стоит заметить, что при наличии ветвений в цикле входящий вес не может быть более чем $d^(n) * s$ для заданной вершины. Значит, алгоритм сходится.

Adaptive Power Method

Описание Алгоритм отличается от предыдущего лишь критерием остановки: пересчет останавливается не для всего вектора сразу, а для отдельных компонент, когда разница между значениями достаточно маленькой или:

$|\pi_k^i - \pi_{k-1}^i| < \epsilon$, где ϵ - заранее заданная точность вычислений.

Как только пересчет всех компонент вектора был остановлен, алгоримт заканчивает выполняться.

Реализация Алгоритм реализован классом AdaptivePowerMethod. При создании объекта класса необходимо передать TransitionMatrix. Необязательным параметром является стартовый вектор весов, который по умолчанию задается как вектор едениц. Коэффициент затухания также может быть передан.

Реализация поддерживает оба формата запуска.

Сходимость] Аналогично Power Method.

Extrapotaling Adaptive Power Method

Описание Алгоритм является улучшением предыдущего. Раз в L шагов алгоритм высчитывает уточняющий коэффициент: $\lambda_2^k \gamma_2 y_2^T = \frac{(\pi^{(k+1)T} - \pi^{(k)T})^2}{\pi^{(k+2)T} - 2\pi^{(k+1)T} - \pi^{(k)T}}$. Вычисленный коэффициент вычитается из $\pi^{(k)T}$, давая улучшенное приближение вектора весов.

$\pi^{(k+3)T} = \pi^{(k)T} - \lambda_2^k \gamma_2 y_2^T$

Реализация Алгоритм реализован классом ExtrapolatingAdaptivePowerMetho При создании объекта класса необходимо передать TransitionMatrix. Необязательным параметром является стартовый вектор весов, который по умолчанию задается как вектор едениц. Коэффициент затухания также может быть передан.

Реализация поддерживает оба формата запуска, при этом можно задать период экстраполяции - L (10 по умолчанию, должен быть больше 3-х).

2.4 Алгоритмы Монте Карло

Алгоритмы обладют нестабильной сходимостью, что не позволяет правильно использовать их с базовым критерием остановки, при этом они обладают рядом полезных свойств, которые можно использовать.

Endpoint Random start Monte Carlo Method

Описание Задан список ссылок S, где $S[i]$ - список страниц, на которые есть ссылки с страницы i, то есть:

$if j \in S[i]$, то есть ссылка с i страницы на страницу j.

Симуляция прохода - симуляция перемещения пользователя по страницам, при этом:

1) Переход пользователя на любую страницу, на которую имеется ссылка, равновероятен.

2) Переход пользователя по страницам заканчивается с вероятностью $1 - d$

Метод выполняет n или n^2 симуляций проходов, *начинающихся со случайной страницы*, за одну итерацию. Страница, на которой симуляция заканчивается увеличивает свой вес на 1.

Проход:

1. Выбрать случайную страницу i: $i \in [0, N-1]$

2. С вероятностью:

   d - переназначить значение i на случайное j: $j \in L_i$, L_i - множество ссылок с страницы i.

   $(1-d)$ - закончить проход, увеличив ранг страницы i на 1.

Реализация Алгоритм реализован классом EndpointRandomStartMonteCarloM
При создании объекта класса необходимо передать TransitionList. Коэффициент затухания может быть передан.

Реализация поддерживает только $iterating\_run$, предоставляя на выбор два возможных количества прогонов за итерацию:

linear - n,

square - n^2, где n - количество страниц.

Оценка сходимости $E(\hat{\pi}_j, N) = \pi_j$

$Var(\hat{\pi}_j, N) = N^{-1}\pi_j(1-\pi_j) < 1/(4N)$

Endpoint Cyclic Start Monte Carlo Method

Описание Задан список ссылок S, где $S[i]$ - список страниц, на которые есть ссылки с страницы i, то есть:

$if j \in S[i]$, то есть ссылка с i страницы на страницу j.

Симуляция прохода - симуляция перемещения пользователя по страницам, при этом:

1) Переход пользователя на любую страницу, на которую имеется ссылка, равновероятен.

2) Переход пользователя по страницам заканчивается с вероятностью $1 - d$

Метод выполняет *заданное количество симуляций проходов для каждой страницы* за одну итерацию. Страница, на которой симуляция заканчивается увеличивает свой вес на 1.

Проход:

1. Назначить i заданное значение: $i \in [0, N - 1]$

2. С вероятностью:

   d - переназначить значение i на случайное j: $j \in L_i$, L_i - множество ссылок с страницы i.

   $(1 - d)$ - закончить проход, увеличив ранг страницы i на 1.

Реализация Алгоритм реализован классом EndpointCyclicStartMonteCarloMe
При создании объекта класса необходимо передать TransitionList. Коэффициент затухания может быть передан.

Реализация поддерживает только *iterating_run*, предоставляя

возможность выбора количества запусков симуляций прохода с каждой страницы - m (3 по умолчанию).

Оценка сходимости $E(\hat{\hat{\pi}}_j) = \pi_j$
$Var(\hat{\hat{\pi}}_j) = N^{-1}[\pi_j - n^{-1}\sum_{i=1}^{n} p_{ij}^2] < Var(\hat{\pi}_j, N)$

Complete Path Monte Carlo Method

Описание Задан список ссылок S, где $S[i]$ - список страниц, на которые есть ссылки с страницы i, то есть:

$if j \in S[i]$, то существует ссылка с i страницы на страницу j.

Симуляция прохода - симуляция перемещения пользователя по страницам, при этом:

1) Переход пользователя на любую страницу, на которую имеется ссылка, равновероятен.

2) Переход пользователя по страницам заканчивается с вероятностью $1 - d$

Метод выполняет заданное количество симуляций проходов для каждой страницы за одну итерацию. *Каждая посещенная в процессе симуляции страница* увеличивает свой вес на 1.

Проход:

1. Назначить i заданное значение: $i \in [0, N-1]$

2. С вероятностью:

   d - увеличить значение ранга страницы i на 1, переназначить значение i на случайное j: $j \in L_i$, L_i - множество ссылок с страницы i.

   $(1 - d)$ - закончить проход, увеличив ранг страницы i на 1.

Реализация Алгоритм реализован классом CompletePathMonteCarloMethod. При создании объекта класса необходимо передать TransitionList. Коэффициент затухания может быть передан.

Реализация поддерживает только $iterating\_run$, предоставляя возможность выбора количества запусков симуляций прохода с каждой страницы - m (3 по умолчанию).

Stopping Complete Path Monte Carlo Method

Описание Задан список ссылок S, где $S[i]$ - список страниц, на которые есть ссылки с страницы i, то есть:

$if j \in S[i]$, то есть ссылка с i страницы на страницу j.

Симуляция прохода - симуляция перемещения пользователя по страницам, при этом:

1) Переход пользователя на любую страницу, на которую имеется ссылка, равновероятен.

2) Переход пользователя по страницам заканчивается с вероятностью $1 - d$ *или при попадании на страницу без внешних ссылок.*

Метод выполняет заданное количество симуляций проходов для каждой страницы за одну итерацию. Каждая посещенная в процессе симуляции страница увеличивает свой вес на 1.

Проход:

1. Назначить i заданное значение: $i \in [0, N-1]$
2. Если страница не имеет внешних ссылок, то закончить проход, увеличив ранг страницы i на 1.
3. С вероятностью:

   d - увеличить значение ранга страницы i на 1, переназначить значение i на случайное j: $j \in L_i$, L_i - множество ссылок с страницы i.

   $(1 - d)$ - закончить проход, увеличив ранг страницы i на 1.

Реализация Алгоритм реализован классом StoppingCompletePathMonteCarlo
При создании объекта класса необходимо передать TransitionList. Коэффициент затухания может быть передан.

Реализация поддерживает только $iterating\_run$, предоставляя

возможность выбора количества запусков симуляций прохода с каждой страницы - m (3 по умолчанию).

Метод выполняет заданное число симуляций проходов для каждой страницы, завершающихся с вероятностью $(1-d)$ или при попадании на страницу без внешних ссылок. Все страницы, посещенные в процессе прохода, увеличивают показатель ранга на 1. Возвращает итоговый вектор сумм.

Random start Stopping Complete Path Monte Carlo Method

Описание Задан список ссылок S, где $S[i]$ - список страниц, на которые есть ссылки с страницы i, то есть:

$if j \in S[i]$, то есть ссылка с i страницы на страницу j.

Симуляция прохода - симуляция перемещения пользователя по страницам, при этом:

1) Переход пользователя на любую страницу, на которую имеется ссылка, равновероятен.

2) Переход пользователя по страницам заканчивается с вероятностью $1-d$ *или при попадании на страницу без внешних ссылок.*

Метод выполняет n или n^2 симуляций проходов, *начинающихся со случайной страницы*, за одну итерацию. Каждая посещенная в процессе симуляции страница увеличивает свой вес на 1.

Проход:

1. Выбрать случайную страницу i: $i \in [0, N-1]$
2. Если страница не имеет внешних ссылок, то закончить проход, увеличив ранг страницы i на 1.
3. С вероятностью:

   d - увеличить значение ранга страницы i на 1, переназначить значение i на случайное j: $j \in L_i$, L_i - множество ссылок с страницы i.

   $(1-d)$ - закончить проход, увеличив ранг страницы i на 1.

Реализация Алгоритм реализован классом RandomStartStoppingCompletePat
При создании объекта класса необходимо передать TransitionList. Коэффициент затухания может быть передан.

Реализация поддерживает только $iterating\_run$, предоставляя на выбор два возможных количества прогонов за итерацию:

linear - n,

square - n^2, где n - количество страниц.

Глава 3

Генераторы

3.1 BAmodel

Описание BAmodel - генератор, основанный на модели Барабаши-Альберта и preferential attachment алгоритме. Данный генератор создает граф со свойствами графа интернета, что позволяет получать хорошие примеры для анализа алгоритмов.

Генератор работает следующим образом:

1. Создается вершина, для которой вычисляется ранг. Формула вычисления ранга: $r_i = 1 - d + d * \sum W_{i,j}$, где $W_{i,j}$ - вес ссылки из j в i.

2. Выбирается множество S - состоящее из 1-3 вершин, выбран-

ных из созданных ранее. Вершина выбирается с вероятностью P_i, пропорциональной количеству входящих ссылок, то есть:

$P_i = \frac{L_{in}}{L_{all}}$, где L_{in} - количество входящих в страницу i ссылок, а L_{all} - общее количество внешних ссылок.

3. Создается новая вершина, имеющая ссылки на вершины из множества S.

4. Вызывается пересчет ранга для новой страницы. При вызове пересчета рангов у любой страницы автоматически вызывается пересчет у всех страниц, на которые та ссылается.

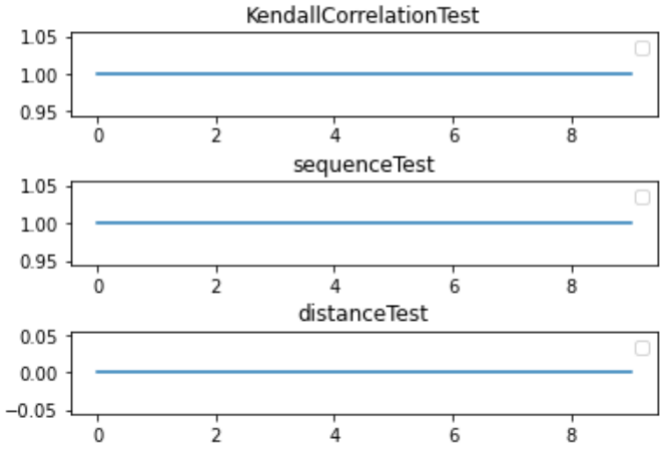
5. Если количество страниц меньше заданного, переход к пункту 2.

Стоит заметить, что данный алгоритм генерирует граф, в котором отсутствуют циклы. Это позволяет утверждать, что вес вершины будет пересчитываться со сложностью пропорциональной диаметру графа $O(DIAM)$ При этом, согласно свойству графа $DIAM$ будет меньше некоторой константы (для графа всего интернета $DIAM < 7$).

Тестирование *Тестирование данного генератора выполнялось после тестирования Power Method.*

Для тестирования данного метода был выбран Power Method, как наиболее точный алгоритм. В данном тестировании генерируются небольшие графы интернета (10 страниц), при этом точность алгоритма выбирается заведомо чрезмерно большой для данного графа. Это делается с целью минимизации ошибки в ранжировании страниц. Полученные в результате работы алгоритмов вектора сравниваются по трем тестам: KendallCorrelationTest, sequenceTest, topTest. Данное тестирование выполняется 100 раз и отрисовывается в виде графиков, где по оси x - итерация, по оси y - значение теста.

```
Mean KendallCorrelationTest: 1.0 should be: 1
Mean sequenceTest: 1.0 should be: 1
Mean distanceTest: 0.0 should be: 0
```



Как видно из графиков и средних значений, BAmodel выполнил верный пересчет рангов для всех сгенерированных графов.

Результаты и код находятся в файле imoc.html: приложение 2.

Глава 4

Тесты

4.1 Описание тестов

Все тесты (кроме Level Test) используют заранее подготовленные файлы с известными ответами. Первый и третий файл - примеры, найденные в интернете. Второй был сгенерирован с помощью SeparatedLevelsTreeGe результат на нем просчитан с помощью Power Method. Четвертый файл теста - смоделированный алгоритмом Барабаши-Альберта граф интернета с заранее известным вектором рангов.

Position Test

Описание Тест создает списки страниц, упорядоченные по убыванию ранга, для истинного и полученного результатов. После этого считает количество совпадений элементов, расположенных на местах с одинаковым номером и возвращает процент совпадений:

Position Test = $\frac{\sum_{i=0}^{N} I(r_i=t_i)}{N}$, где

N - количество страниц,

I - индикатор выражения, возвращает 1, если выражение истинно, и 0, если выражение ложно,

r_i - i-я по рангу страница, изначальный список ранжирования есть результат работы алгоритма.

t_i - i-я по рангу страница, изначальный список есть истинный список ранжирования.

Тест принимает значение от 0 до 1, где 1 - абсолютное совпадение, 0 - абсолютное несовпадение.

Оценка метрики Тест оценивает простой и понятный параметр, который легко интерпретировать. Но простота этого теста также его главный недостаток. Ряд ситуаций могут быть неправильно интерпертированы, если рассматривать их со стороны этого теста, например:

1) Когда одна или несколько страниц оказались не на своей позиции, все сдвинутые страницы будут расценены как ошибочно опре-

деленные.

2) Алгоритм, который хорошо определяет страницы с наибольшими рангами, будет иметь меньшее значение теста чем алгоритм, который работает лучше в среднем, хотя определение первых страниц является более важной задачей.

Sequence Test

Описание Тест создает списки страниц, упорядоченные по убыванию ранга, для истинного и полученного результатов. После этого считает количество совпадений элементов, расположенных на местах с одинаковым номером. Если номера страниц не совпадают, то страница, стоящая в списке полученного результата, удаляется.

Sequence Test $= \frac{D)}{N}$, где

N - количество страниц,

D - количество страниц, после удаления всех несовпадающих страниц.

Тест принимает значение от 0 до 1, где 1 - абсолютное совпадение, 0 - абсолютное несовпадение.

Оценка метрики Тест является улучшением предыдущего. Данные изменения позволяют получить более правильную оценку в 1

ситуации предшествующего теста. Так же тест сглаживает неточности и проверяет более процент сопадения последовательностей чем процент полного совпадения. Но при этом:

1) Метрика не дает серьезных улучшений для случая 2 предыдущего алгоритма.

2) Метрика слабо отражает выбросы. Если страница с наименьшим рангом алгоритмом определится первой, при этом N = 10000 и все остальные страницы определятся правильно, то ошибка составит всего $1/N = 0.0001$.

Vector Test

Описание Тест нормализует истинный и полученный списки рангов, после чего возвращает расстояние между векторами по метрике l1.

Vector Test = $\sum_{i=0}^{N} |R_i - T_i|$, где

N - количество страниц,

R_i - i-я по рангу страница, изначальный список ранжирования есть результат работы алгоритма.

T_i - i-я по рангу страница, изначальный список есть истинный список ранжирования.

Тест принимает значение 0, когда вектора полностью совпали, и

2, когда вектора максимальное расстояние по l1.

Замечание Так как вектора нормализованны, то

$\sum_{i=0}^{N} R_i = 1$, $\sum_{i=0}^{N} T_i = 1$, тогда:

1) если для любой пары (R_i, T_i) выполняется $R_i = 0$ or $T_i = 0$, то $\sum_{i=0}^{N} |R_i - T_i| = \sum |R_k| + \sum |T_j| = 1 + 1 = 2$ где $i \neq j$

2) если для пар из множества Z это свойство не выполнено, то для всех пар со свойством их сумма $\sum |Rz_i - Tz_i| = \sum |Rz_k| + \sum |Tz_j|$. $\sum_{z \in Z} |R_z - T_z| = c - \delta$, где c - сумма всех компонент пар $z \in Z$, а δ - сумма минимумов для каждой пары $z \in Z$. Тогда:

$\sum_{i=0}^{N} |R_i - T_i| = \sum |Rz_i - Tz_i| + \sum_{z \in Z} |R_z - T_z| = \sum |Rz_k| + \sum |Tz_j| + c - \delta = (\sum |Rz_k| + \sum |Tz_j| + c) - \delta = \sum |R_k| + \sum |T_j| - \delta < \sum |R_k| + \sum |T_j| = 2$

Значит, 2 - максимальное значение данного теста.

Оценка метрики Метрика оценивает покомпонентное расстояние между векторами. Данное значение просто интерпретировать, метрика хорошо определяет некоторые выбросы, например, когда у какой-то страницы сильно завышен или занижен ранг, а также ситуации, когда ранги достаточно сильно схожи. При этом метрика слабо выявляет ошибки первых страниц, например, если вектора в среднем похожи.

Distance Test

Описание Тест считает среднее расстояние страниц от их истинного положения, то есть:

Distance Test $= \frac{\sum_{i=0}^{N} |RP[i]-TP[i]|}{N}$, где

N - количество страниц,

$RP[i]$ - место i-той страницы в списке ранжирования алгоритма.

$TP[i]$ - место i-той страницы в истинном списке ранжирования.

Тест принимает значение 0, когда вектора полностью совпали. Любое значение больше означает наличие различий.

Оценка метрики Метрика служит для определения степени ошибки алгоритма в среднем и работает больше как дополнение к Sequence Test, позволяя лучшее анализировать и понимать результаты. Как самостоятельная метрика имеет множество недостатков: зависимость от выбросов, качество не зависит от места ошибок, зависимость от сдвига страниц.

Level Test

Описание Тест использует BinaryTreeGenerator, для генерации графа, который являлся бы бинарным деревом. Для данного дере-

ва нам наперёд известны позиции страниц в ранговом списке. Тест возвращает процент страниц с правильно определённым уровнем.

Level Test = $\frac{\sum_{i=0}^{N} I(RL[i]=TL[i])}{N}$, где

N - количество страниц,

I - индикатор выражения, возвращает 1, если выражение истинно, и 0, если выражение ложно,

$RL[i]$ - уровень i-й по рангу страницы, изначальный список ранжирования есть результат работы алгоритма.

$TL[i]$ - уровень i-й по рангу страницы, изначальный список есть истинный список ранжирования.

Тест принимает значение от 0 до 1, где 1 - абсолютное совпадение, 0 - абсолютное несовпадение.

Оценка метрики Тест является по сути оценкой разумности для всех алгоритмов, чей результат не является случайным. Так как любое значение кроме 1 будет означать наличие ошибки в работе алгоритма.

Для случайных алгоритмов является показателем работы с простым случаем, а так же неплохой проверкой того, как алгоритм оценивает страницы с одинаковыми рангами.

Top Test

Описание Тест считает процент совпадений первых i страниц, для всех i меньших заданного числа, и возвращает список процентов сопадений для каждого i, то есть:

Top Test [j] = $\frac{\sum_{i=0}^{j} I(RS[0{:}j][i] \in TS[0{:}j])}{j}$, где

j - количество первых страниц,

I - индикатор выражения, возвращает 1, если выражение истинно, и 0, если выражение ложно,

$RS[0:j][i]$ - i-я страница в срезе, изначальный список ранжирования есть результат работы алгоритма.

$TS[0:j]$ - множество первых j страниц в истинном списке ранжирования.

j-я ячейка теста принимает значение от 0 до 1, где 1 - абсолютное совпадение множеств, 0 - абсолютное несовпадение.

Оценка метрики Метрика является отличным параметром качества определения топа страниц, то есть страниц, имеющих наибольшие ранги. Это очень важный параметр, так как пользователь чаще всего переходит по странице, находящейся в первой пятёрке. При этом тест не может быть использован отдельно от остальных метрик, так как сосвем не оценивает правильность определения рангов на для остальных страниц.

Порог страниц данного теста не следует брать слишком большим,

так как качество результатов данного теста напрямую зависит от количества элементов, например:

Если алгоритм выдаст обратное ранжирование, то каждая последующая компонента теста будет все ближе подходить к 1, но это не будет означать, что алгоритм выдает хороший ответ.

Kendall Test

Описание Заданные вектора нормализуются, а затем для них считается корреляция Кендалла.

Оценка метрики Тест оценивает совпадение последовательностей, при этом имеет схожие с Sequence Test недостатки, но при этом эти ошибки имеют куда менее заметный эффект. За это сглаживание приходится платить понятностью, метрика немного более сложная в интерпретации.

4.2 Запуск тестов

Чтобы запустить тесты, достаточно запустить метод CompleteTest() из модуля Test, то есть выполнить следующий код:

```
# Python
```

```
import PageRank.Test as t
t.CompleteTest()
```

Результат тестов появится в файле CompleteTestResults.html

4.3 Результаты алгоритмов на тестах

Итеративные алгоритмы Все итеративные алгоритмы выдают полностью правильные ответы на примерах (первый и третий тесты), при этом на втором тесте в ответах появляются расхождения из-за специфики работы тех или иных алгоритмов, но при этом лидирующие страницы определяются точно. Исключением является цепь Маркова, она имеет серьезные расхождения с Power Method и его улучшениями, что, вероятно, связано с плохо подходящим в этом тесте базвым показателем точности.

Алгоритмы Монте Карло

Endpoint Random Algorithms Алгоритмы данной группы показывают неплохие результаты на примерах, на втором тесте показатели хуже, но топ станиц определен достаточно точно, при этом они отработали примерно в 6 раз быстрее Extrapotaling Adaptive Power Method.

Complete Path Algorithms Алгоритмы показывают не очень хорошие значения части тестов, что частично связано с малым заданным числом проходов (алгоритмы в тестах настроены на небольшое число страниц). Имеют преимущества предыдущей группы алгоритмов: скорость, высокое качество определения лидирующих страниц, но имеют лучшие значения тестов. В этой группе выделяется Stopping Complete Path Monte Carlo Method, так как все страницы без ссылок на них имеют одинаковое значение ранга.

Глава 5

Сравнение алгоритмов

5.1 Описание методологии сравнения

В качестве базы для сравнения была выбрана точность, достижимая за некоторое время. Для тестирования используется функция AccurasyOverTime, принимающая на вход метрику и список значений времени. Каждый алгоритм работает время не меньшее чем заданное в списке значений. После чего по полученным данным времени и точности строятся графики.

В приложении приведены графики для четырех метрик: sequenceTest, distanceTest, KendallCorrelationTest, topTestNumber. topTestNumber - адаптированный алгоритм topTest. Возвращает среднее значение полученного от topTest списка.

Результаты и код находятся в файле imoc.html: приложение 1.

5.2 Итоги сравнения

sequenceTest, distanceTest, KendallCorrelationTest Алгоритмы группы Power имеют наилучшие показатели на этих тестах, MarkovChain также выдает достаточно качественные результаты, но уже ощущается нехватка времени. Из алгоритмов Monte Carlo явно выделяется StoppingCompletePathMonteCarloMethod, имеющий результаты сильно лучше других из этой группы.

topTestNumber При заданном списке значений времени все алгоритмы легко определяют ведущие 5 страниц. Исключение составляет лишь MarkovChain алгоритм, результаты которого страдают от нехватки времени.

После данных экспериментов может показаться, что использование алгоритмов Monte Carlo нецелесообразно, так как при одинаковых затратах времени они сильно проигрывают в точности. Однако, обратимся к следующему эксперименту.

Small topTestNumber В данном тесте алгоритмы ставятся в очень узкие рамки времени: промежутки столь малы, что итеративные ал-

горитмы не успевают провести даже одну итерацию. В то же время алгоритмы Monte Carlo на графе из 500 страниц определяют первые пять страниц с точностью не меньшей чем у итеративных алгоритмов и за время в несколько раз меньшее.

Замечание Данная методика тестирования никак не использует преимущества AdaptivePowerMethod, так как критерий остановки не используется вовсе.