UB Social

CS50AI with Python

5.PageRank

Mateus Schwede

Problemática

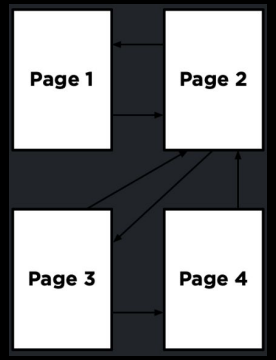
- Escreva IA para classificar páginas da web por importância.

```
$ python pagerank.py corpus0
PageRank Results from Sampling (n = 10000)
  1.html: 0.2223
  2.html: 0.4303
  3.html: 0.2145
  4.html: 0.1329
PageRank Results from Iteration
  1.html: 0.2202
  2.html: 0.4289
  3.html: 0.2202
  4.html: 0.1307
```

Problemática

- Quando mecanismos de busca como Google exibe resultados de pesquisa, colocam-se páginas mais "importantes" e de maior qualidade mais acima nos resultados de pesquisa do que páginas menos importantes;
- Uma heurística pode definir página como "importante" é quando muitas outras páginas têm links, já que é razoável imaginar que mais sites terão links para página da web de maior qualidade do que uma de menor qualidade. Pode-se, portanto, imaginar sistema em que cada página recebe classificação conforme nº de links de entrada que ela tem de outras páginas, e classificações mais altas sinalizam maior importância;
- Contudo, se alguém quiser fazer com que sua página pareça mais importante, então, sob este sistema, pode-se simplesmente criar muitas outras páginas com links para a página desejada, inflando artificialmente sua classificação;
- Por isso, algoritmo PageRank foi criado pelos cofundadores do Google, onde site é mais importante se ele for vinculado a outros sites importantes, e links de sites menos importantes têm seus links ponderados menos.

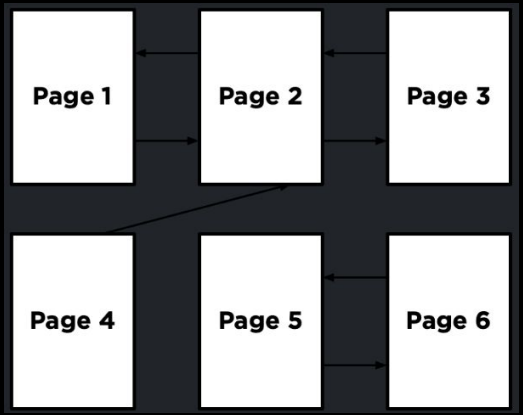
Random Surfer Model



- PageRank pode ser pensado via random surfer model, que considera comportamento de surfer hipotético na internet que clica em links aleatoriamente. Considere corpus de páginas da web, onde seta entre 2 páginas indica link de página para outra;
- Random surfer considera surfer que começa com página da web aleatoriamente e, então, escolhe aleatoriamente links para seguir. Se surfer estiver na Página 2, por exemplo, escolheria aleatoriamente entre Páginas 1 e 3 para visitar em seguida (links duplicados na mesma página são tratados como único link, e links de página para si mesma também são ignorados). Se escolhesse Página 3, surfer escolheria aleatoriamente entre Páginas 2 e 4 para visitar em seguida;
- PageRank de página, então, pode ser descrito como probabilidade de que surfer aleatório esteja naquela página em dado momento. Afinal, se houver mais links para página específica, então é mais provável que surfer aleatório acabe naquela página. Além disso, link de site mais importante tem mais probabilidade de ser clicado do que de site menos importante para o qual há menos páginas vinculadas, onde este modelo também lida com links de ponderação por sua importância;
- Uma forma de interpretar o modelo é como Cadeia de Markov (Markov Chain), onde cada página representa estado, e cada página tem modelo de transição que escolhe entre seus links aleatoriamente. A cada passo de tempo, estado muda para uma das páginas vinculadas pelo estado atual;

Random Surfer Model

- Ao amostrar estados aleatoriamente da Cadeia de Markov, obtém-se estimativa para PageRank de cada página. Inicia-se escolhendo página aleatoriamente e, em seguida, continua seguindo links aleatoriamente, mantendo controle de quantas vezes visita-se cada página. Após reunir todas nossas amostras (com base em nº escolhido com antecedência), proporção do tempo em que estive-se em cada página pode ser estimativa para rank da página;
- Porém, iniciando aleatoriamente amostrando Página 5. Não há escolha senão ir para Página 6, então não há escolha senão ir para Página 5 depois disso, então para Página 6 novamente, e assim por diante. Acaba-se com estimativa 0,5 para PageRank das Páginas 5 e 6, e estimativa de 0 para PageRank de todas restantes, já que passa-se todo tempo nas Páginas 5 e 6 e nunca visitou-se nenhuma outra;



Random Surfer Model

- Para garantir acesso a outros locais no corpus de páginas da web, usa-se, no modelo, damping factor d. Com probabilidade d (onde d é geralmente definido em torno de 0,85), surfer escolhe link na página atual aleatoriamente. Mas, caso contrário (com probabilidade 1 - d), surfer escolhe uma de todas páginas no corpus aleatoriamente (incluindo aquela em que está);
- Surfer agora começa escolhendo página aleatoriamente e, então, para cada amostra adicional para gerar, escolhe link da página atual aleatoriamente com probabilidade de escolher qualquer página aleatoriamente com probabilidade 1 - d. Se mantiver controle de quantas vezes cada página apareceu como amostra, pode-se tratar proporção de estados que estava em determinada página como seu PageRank.

Iterative Algorithm

- Também pode-se definir PageRank de página usando expressão matemática recursiva. Seja PR(p) PageRank de determinada página p: probabilidade de surfer acabar naquela página. Como definimos PR(p)? Há 2 maneiras de surfer acabar na página;
    a. Probabilidade 1 - d, internauta escolheu página aleatoriamente e acabou na página p;
    b. Probabilidade d, internauta navegou link de página i para página p.
- Condição 1 é direta de expressar matematicamente: é 1 - d dividido por N, onde N é nº total de páginas em corpus. Isso ocorre porque probabilidade 1 - d de escolher página aleatoriamente é dividida uniformemente entre todas N páginas possíveis;
- Na 2ª condição, considera-se cada página possível i que faz link para página p. Para cada página de entrada, deixe NumLinks(i) ser nº de links na página i. Cada página i que faz link para p tem seu próprio PageRank, PR(i), representando probabilidade de estar na página i em qualquer momento. E como da página i viaja-se para qualquer link nesta página com probabilidade igual, divide-se PR(i) pelo nº de links NumLinks(i) para obter probabilidade de estar na página i e escolher link para página p;

$$PR(p) = \frac{1-d}{N} + d \sum_{i} \frac{PR(i)}{NumLinks(i)}$$

Iterative Algorithm

- Na fórmula, d é fator de amortecimento (damping factor), N é nº total de páginas no corpus, i abrange todas páginas com links para página p e NumLinks(i) é nº de links presentes na página i;
- Calcula-se valores de PageRank de cada página via iteração, assumindo que PageRank de cada página é 1 / N. Após, usa-se fórmula para calcular novos valores de PageRank para cada página, conforme valores anteriores de PageRank. Repetindo tal processo, calcula-se novo conjunto de valores de PageRank para cada página baseado no conjunto anterior de valores de PageRank, eventualmente valores de PageRank convergirão (ou seja, não mudarão em mais do que pequeno limite a cada iteração);
- No projeto, implemente ambas abordagens para calcular PageRank - tanto por amostragem de páginas de surfer da Cadeia de Markov quanto pela aplicação iterativa da fórmula do PageRank.

Instruções

- Baixe código de https://cdn.cs50.net/ai/2023/x/projects/2/pagerank.zip e descompacte-o.

Funcionamento

- Em pagerank.py, define-se 2 constantes: DAMPING representa damping factor, inicialmente definido como 0,85. SAMPLES representa nº de amostras que serão usadas para estimar PageRank via método de amostragem, inicialmente definido como 10.000 amostras;
- Função main recebe argumento de linha de comando, nome de diretório de corpus de páginas da web para calcular PageRanks. Função crawl usa esse diretório, analisa todos arquivos HTML no diretório e retorna dicionário representando corpus. As chaves nesse dicionário representam páginas (por exemplo, "2.html"), e valores do dicionário são conjunto de todas páginas vinculadas pela chave (por exemplo, {"1.html", "3.html"});
- Função main chama função sample_pagerank, cujo propósito é estimar PageRank de cada página por amostragem, tomando como argumentos corpus de páginas geradas pelo crawl, damping_factor e nº de amostras a serem usadas. Por fim, sample_pagerank retorna dicionário onde chaves são cada nome de página e valores são PageRank estimado de cada página (nº entre 0 e 1);
- Função main chama função iterate_pagerank, que também calculará PageRank para cada página, via método de fórmula iterativa em vez de por amostragem. Espera-se que valor de retorno esteja no mesmo formato, e espera-se que saída das 2 funções seja semelhante dado mesmo corpus.

Especificações

- Conclua implementação de transition_model, sample_pagerank e iterate_pagerank;
- transition_model deve retornar dicionário representando distribuição de probabilidade sobre qual página internauta visitaria em seguida, dado corpus de páginas, página atual e damping factor. Função aceita 3 argumentos: corpus, page e damping_factor:
    - corpus é dicionário Python que mapeia nome de página para conjunto de todas páginas vinculadas por essa página;
    - page é string que representa em qual página internauta está no momento;
    - damping_factor é nº float que representa damping factor a ser usado ao gerar probabilidades.
  - Valor de retorno da função deve ser dicionário Python com chave para cada página no corpus. Cada chave deve ser mapeada para valor que representa probabilidade de que surfer escolheria essa página em seguida. Valores nessa distribuição de probabilidade retornada devem somar 1:
    - Com damping_factor de probabilidade, surfer deve escolher aleatoriamente link da página com igual probabilidade;
    - Com probabilidade 1 - damping_factor, internauta deve escolher aleatoriamente uma das páginas do corpus com igual probabilidade.

Especificações

- Exemplo, corpus {"1.html": {"2.html", "3.html"}, "2.html": {"3.html"}, "3.html": {"2.html"}}, página "1.html" e damping_factor 0,85, então saída do transition_model será {"1.html": 0,05, "2.html": 0,475, "3.html": 0,475}. Isso ocorre porque com probabilidade 0,85, escolhe-se aleatoriamente ir da página 1 para 2 ou 3 (então cada página 2 ou 3 tem probabilidade 0,425 para começar), mas cada página recebe 0,05 adicional porque com probabilidade 0,15 escolhe-se aleatoriamente entre todas 3 páginas;
- Se página não tiver links de saída, então transition_model deve retornar distribuição de probabilidade que escolhe aleatoriamente entre páginas com probabilidade igual;

- Função sample_pagerank aceita corpus de páginas da web, damping factor e nº de amostras, retornando PageRank estimado para cada página;
- Função aceita 3 argumentos: corpus, a damping_factor e n:
  - corpus é dicionário Python que mapeia nome de página para conjunto das páginas vinculadas por essa;
  - damping_factor é nº float que representa damping factor a ser usado pelo modelo de transição;
  - n é nº inteiro que representa nº de amostras que devem ser geradas para estimar valores do PageRank.

Especificações

- Valor de retorno da função deve ser dicionário Python com chave para cada página no corpus. Cada chave deve ser mapeada para valor que representa PageRank estimado daquela página (ou seja, a proporção de todas amostras que correspondem àquela página). Valores neste dicionário devem somar 1;
- 1ª amostra deve ser gerada escolhendo aleatoriamente uma página;
- Para cada amostras restante, próxima amostra deve ser gerada a partir da anterior com base no modelo de transição de tal:
  - Passar amostra anterior para função transition_model, junto com corpus e damping_factor, para obter probabilidades para próxima amostra;
  - Exemplo, probabilidades de transição forem {"1.html": 0,05, "2.html": 0,475, "3.html": 0,475}, então 5% das vezes próxima amostra gerada deverá ser "1.html", 47,5% das vezes próxima amostra gerada deverá ser "2.html" e 47,5% das vezes próxima amostra gerada deverá ser "3.html".
- Pode-se assumir que n será pelo menos 1;

Especificações

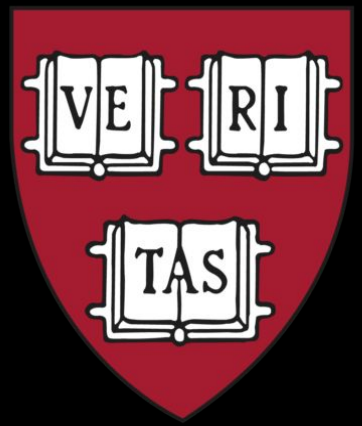
- Função iterate_pagerank deve aceitar corpus de páginas da web e damping factor, calcular PageRanks com base na fórmula de iteração e retornar PageRank de cada página com precisão de 0,001;
    - Função aceita 2 argumentos: corpus e damping_factor:
        - Corpus é dicionário Python que mapeia nome de página para conjunto das páginas vinculadas por essa página;
        - damping_factor é nº float que representa fator de amortecimento a ser usado na fórmula do PageRank.
    - Valor de retorno da função deve ser dicionário Python com chave para cada página no corpus. Cada chave deve ser mapeada para valor que representa PageRank daquela página. Valores neste dicionário devem somar 1;
    - Função deve começar atribuindo a cada página classificação de 1 / N, onde N é nº total de páginas no corpus;
    - Função deve calcular repetidamente novos valores de classificação com base em nos valores de classificação atuais, conforme fórmula PageRank na seção "Background" (ou seja, calcular PageRank de página com base nos PageRanks das páginas que têm links para ela);

Especificações

        - Página que não possui nenhum link deve ser interpretada como tendo 1 link para cada página do corpus (incluindo ela mesma);
    - O processo deve ser repetido até que nenhum valor de PageRank seja alterado em mais de 0,001 entre valores de classificação atuais e novos valores de classificação;
- Não é permitido modificar nada mais em pagerank.py além das 3 funções que especificação solicita para implementar, embora possa escrever funções adicionais e/ou importar outros módulos da biblioteca padrão do Python. Também pode-se importar numpy ou pandas, mas não deve-se usar nenhum outro módulo Python de terceiros.

Submissão

- Visual Studio Code online: https://cs50.dev
- Testar precisão da lógica do algoritmo: check50 ai50/projects/2024/x/pagerank
- Testar estilização do código: style50 pagerank.py
- Para submissão:
  - Em https://submit.cs50.io/invites/d03c31aef1984c29b5e7b268c3a87b7b, entre com GitHub e autorize CS50;
  - Instale pacote Git, Python 3 (e pip), instalando pacotes: pip3 install style50 check50 submit50
  - Submeta o projeto: submit50 ai50/projects/2024/x/pagerank
- Verificar avaliação: https://cs50.me/cs50ai.



UB Social

Mateus Schwede
HBS ID 202400167108 - DCE ID @00963203