Zastosowanie wartości własnych macierzy

Adam Iwanicki

Uniwersytet Warszawski

15 maja 2008

Agenda

Motywacja

- Mamy do czynienia z eksplozją informacji.
- Dokładność wielu systemów przeszukujących pozostawia wiele do życzenia.
- Wiele systemów opartych jest na algebrze liniowej.
- Tradycyjne metody, takie jak LSI (Latent Semantic Indexing) dobre w odniesieniu do małych zbiorów, okazują się praktycznie bezużyteczne w przypadku internetu, ze względu na dużą złożoność.

Jak zbudować wyszikiwarkę?

Wymaga to trzech prostych kroków:

- Trzeba umieć ściągnąć wszystkie dokumenty (albo chociaż dużą część).
- Trzeba umieć przeszukać te dokumenty znajdując te związane z danym zapytaniem.
- Trzeba umieć posortować otrzymaną listę, tak by najwazniejsze dokumenty znalazły się na samej górze.

Dlaczego internet jest trudny do przetwarzania

- Jest ogromny.
- Nieustannie się zmienia. Nowe strony się pojawiają, istniejące się dezaktualizują, inne zostają usunięte.
- Zawiera wiele redundantnych dokumentów, a także wiele niskiej jakości materiałów.
- Wielu użytkowników stara się umyślnie wprowadzić w błąd systemy, tak by ich strony pojawiały się wysoko w wynikach wyszukiwania.

Dlaczego użytkownicy nie ułatwiają tego zadania

- Zapytania często są krótkie i lakoniczne, najczęściej składające się z 1–2 słów.
- Użytkownicy rzadko używają zaawansowanych cech wyszukiwarek.
- Zwykle przeglądane jest 10–20 pierwszych wyników.

Co może pomóc w przetwarzaniu danych zawartych w internecie?

- Unikalna struktura, powiązana linkami (referencjami).
- Jest ona wykorzystywana w 3 najczęściej omawianych metodach:
  - HITS (Hypertext Induced Topic Search)
  - PageRank
  - SALSA

Latent Semantic Indexing

- Opiera się na rozkładzie na wartości własne macierzy, której jeden wymiar stanowią dokumenty, a drugi terminy.
- Umożliwia znalezienie ukrytych związków między terminami.
- Dzięki temu radzi sobie z polisemami i synonimami przypisując dokumenty i terminy do pewnych pojęć.
- Wadą jest to, że nie bierze pod uwagę jakości dokumentów.

Model internetu

Każda strona /dokument w sieci, jest reprezentowana jako wierzchołek w bardzo dużym grafie.

Skierowane krawędzie między wierzchołkami reprezentują referencje między dokumentami.

Pojęcia

Znawca / Autorytet

Znawca jest wierzchołkiem, który posiada wiele linków wchodzących (ma duży stopień wejściowy).

Węzeł

Węzeł jest wierzchołkiem, z którego wychodzi wiele linków (ma duży stopień wyjściowy).

Założenia

Teza

Algorytm HITS opiera się na stwierdzeniu, że dobre węzły wskazują na dobrych znawców i że dobzi znawcy są wskazywani przez dobre węzły.

Algorytm HITS przypisuje każdemu dokumentow dwie oceny — jedna mówiąca o tym na ile jest on dobrym autorytetem, a druga mówiąca o tym czy jest dobrym węzłem.

Obliczenia

Niech i będzie wierzchołkiem odpowiadającym pewnej stronie, wówczas x_i i y_i będą odpowiednio jego rankingiem jako autorytetu i węzła.
Niech E będzie zbiorem wszystkich krawędzi skierowanych w grafie, a e_ij reprezentuje skierowaną krawędź od wierzchołka i do j.
Załóżmy, że na początku wszystkie strony mają przypisaną jakąś inicjalną wartość $x_i^{(0)}$ i $y_i^{(0)}$, wówczas algorytm iteracyjnie poprawia oceny zgodnie ze wzorem:

$$x_i^{(k)} = \sum_{j:e_{ji}\in E} y_j^{(k-1)} \quad i \quad y_i^{(k)} = \sum_{j:e_{ij}\in E} x_j^{(k)}$$

Obliczenia c.d.

Jeśli teraz użyjemy macierzy sąsiedztwa L, możemy te równania zapisać w formie macierzowej:

$$\begin{aligned} \vec{x}^{(k)} &= \mathbf{L}^T \vec{y}^{(k-1)} \\ \vec{y}^{(k)} &= \mathbf{L}\vec{x}^{(k)} \\ k &= k+1 \end{aligned}$$

Jako początkowy wektor można przyjąć: $\vec{y}^{(0)} = e$.

Obliczenia c.d.

Powyższe równanie można rozwinąć o jeden krok więcej otrzymując:

$$\begin{aligned} \vec{x}^{(k)} &= \mathbf{L}^T\mathbf{L}\vec{x}^{(k-1)} \\ \vec{y}^{(k)} &= \mathbf{L}\mathbf{L}^T\vec{y}^{(k-1)} \end{aligned}$$

Równania te wyznaczają iteracyjną metodę obliczania rankingów za pomocą potęgowania. Obie macierze (LL^T i L^TL) są nieujemnieokreślone.
Tak naprawdę wystarczy policzyć tylko jedno równanie iteracyjnie, a drugie z wcześniejszej zależności.

Implementacja

Implementacja składa się z dwóch głównych kroków:

- Utworzeniu grafu sąsiedztwa N powiązanego z zapytaniem.
- Obliczenia rankingów i zaprezentowaniu użytkownikowi dwóch list.

Wady i zalety

- Podwójny ranking — czasem użytkownika interesuje bardziej jeden z nich.
- Redukuje duży problem do małej instancji.
- Jest zależny od zapytania i obliczenia muszą być wykonane na bieżąco.
- Zwłaszcza jeden ranking jest podatny na spam.
- Jest bardzo „lokalny”.
- Możliwe jest odpłynięcie (zboczenie) od tematu.

Powiązania

Algorytm jest silnie związany z badaniami bibliometrycznymi i takimi pojęciami jak:

co-citation

Ma ono miejsce, kiedy oba dokumenty są cytowane przez trzeci dokument.

co-reference

Ma miejsce kiedy dwa dokumenty odnoszą się do tego samego trzeciego dokumentu.

Zostało pokazane, że zachodzą równości: $L^T L = D_{in} + C_{cit}$ i $LL^T = D_{out} + C_{ref}$

Motywacja

- Algorytm PageRank traktuje linki wchodzące, jako swojego rodzaju rekomendacje danego dokumentu, przez autora strony linkującej.
- Jednak linki z dobrych (poważanych) stron powinny wnosić więcej, niż linki ze stron marginalnych.
- W ten sposób każda strona ma przypisaną liczbę (PageRank), która mówi jak ważna jest dana strona.

Obliczenia

Powyższa idea może być zapisana jako:

$$r(P) = \sum_{Q \in B_P} \frac{r(Q)}{|Q|}$$

gdzie:

- $r(P)$ – oznacza wartość PageRank dla strony P
- B_P – jest zbiorem wszystkich stron wskazujących na P
- $|Q|$ – jest liczbą linków wychodzących z Q

Obliczenia c.d.

Obliczenia podobnie jak przy HITSie mogą być prowadzone iteracyjnie, początkowy wektor można przyjąć arbitralnie wszystkim dokumentom przypisując wartość $\frac{1}{n}$.
Niech $\pi_j^T = \left(r_j(P_1), r_j(P_2), ..., r_j(P_n)\right)$ wówczas:

$$\pi_j^T = \pi_{j-1}^T \mathbf{P}$$

a **P** jest macierzą: $p_{ij} = \begin{cases} \frac{1}{|P_i|} & \text{jeśli } P_i \text{ wskazuje na } P_j \\ 0 & wpp \end{cases}$

Model

- Jeśli założymy, że nie ma stron bez linków wychodzących, albo stronom takim arbitralnie dodamy takie linki, to macierz **P** jest macierzą stochastyczną.
- Oznacza to, że każda iteracja obliczania PageRanku jest przejściem łańcuchem Markowa. Łańcuch ten jest losowym błądzeniem po grafie zdefiniowanym przez strukturę linków.
- PageRank odpowiada rozkładowi stacjonarnemu tego łańcucha Markowa, dlatego wartość PageRank mówi o proporcji czasu spędzonego na danej stronie, przy założeniu losowego klikania na linki.

Dostosowywanie macierzy

Jest kilka problemów, które powodują, że nie powinniśmy używać dokładnie macierzy reprezentującej nasz graf:

- Macierz może nie być stochastyczna (niektóre wiersze mogą mieć zerową sumę). Wówczas zamiast takiego wiersza możemy użyć $\frac{e^T}{n}$
- Powstała macierz może być redukowalna (odpowiadający jej łańcuch może być redukowalny — istnieją zbiory stanów, z których nie można się wydostać).

Wady i zalety

- Odpływanie od tematu.
- Niezależność od zapytania.
- Odporność na spam.
- Możliwość personalizacji, przez wektor perturbacji.

Stochastic Approach for Link Structure Analysis

Jest to połączenie idei HITS i PageRanka.
Tak jak w HITSie tworzone są dwa rankingi.
Są one generowane za pomocą łąńcuchów Markowa tak jak w przypadku PageRanka.

Sposób obliczania

- Podobnie jak w przypadku HITSa w czasie zapytania tworzony jest graf „sąsiedztwa”.
- Następnym krokiem zamiast tworzenia macierzy sąsiedztwa jest stworzenia grafu dwódzielnego, którego jedną grupę wierzchołków stanowią dokumenty o niezerowym stopniu wyściowym, a drugą dokumenty o niezerowym stopniu wejściowym. Krawędziemi są odpowiadające im krawędzie skierowane w pierwotnym grafie.

Sposób obliczania c.d.

- Kolejnym krokiem jest obliczenie macierzy prawdopodobieństwa **H** odpowiadającej „węzłowatości” i **A** odpowiadającej autorytarności.
- O ile Google’owa macierz **P** była ważona tylko w wierszach, to w przypadku SALSy mamy dwie macierze sąsiedztwa L_c (L_r), w których każda niezerowa kolumna (wiersz) została podzielona przez swoją sumę.
- Następnie kładziemy:

$$\begin{aligned} \mathbf{H} &= L_r L_c^T \\ \mathbf{A} &= L_c^T L_r \end{aligned}$$

Sposób obliczania c.d.

- Pozostaje już tylko znaleźć rozkłady stacjonarne π_h^T macierzy **H** oraz π_a^T macierzy **A**.
- Jednak w przypadku SALSy z redukowalnością radzimy sobie w inny sposób:
  - zerowe wiersze i kolumny są usuwane,
  - liczymy rozkłady stacjonarne dla spójnych składowych grafu oddzielnie,
  - następnie przeskalowujemy je, aby utworzyły globalny rozkład.

Wady i zalety

- Jest mniej podatny na spam.
- Jest zależny od zapytania.
- Jest mnie podatny na odpływanie od tematu.
- Poprzez podział na spójne składowe ułatwia obliczanie.
- Posiada dwa rankingi.

Bibliografia


Carl D. Meyer, Amy N. Langville,
A Survey of Eigenvector Methods of Web Information Retrieval

- http://en.wikipedia.org/wiki/Latent_semantic_analysis
- http://mathworld.wolfram.com/Eigenvector.html