Nowa punktowa metoda uczenia rankingu / A new pointwise learn-to-rank algorithm

Mikołaj Fejzer, Jakub Narębski, Piotr Przymus, Krzysztof Stencel

14.05.2021

Seminarium badawcze Zakładu Logiki - Wnioskowania aproksymacyjne w eksploracji danych

Wprowadzenie

Ranking wyników jest podstawą

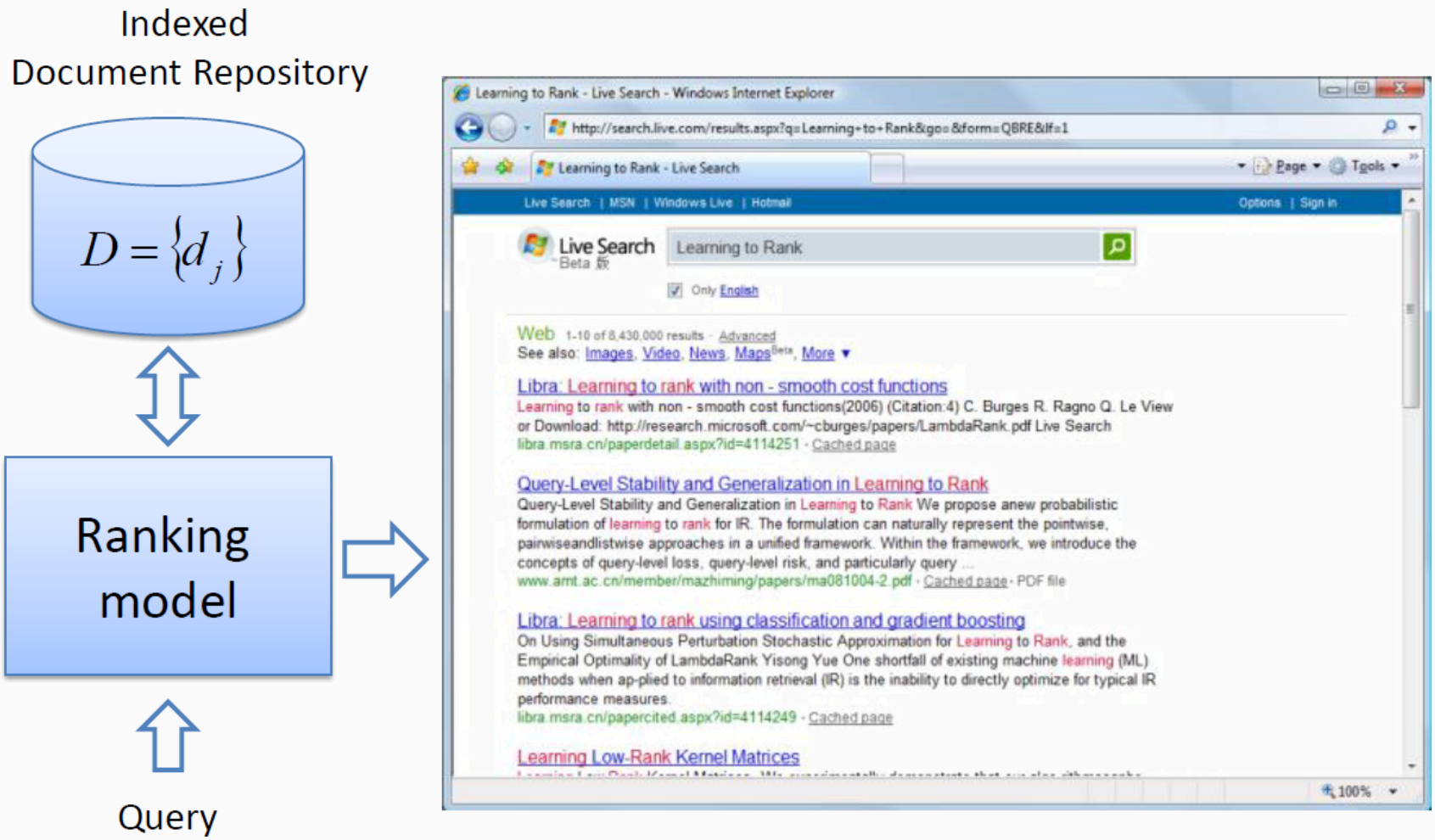

Figure 1: Tie-Yan Liu, “Learning to Rank for Information Retrieval”

Zastosowania rankowania wyników

- Wydobywanie dokumentów
- Systemy rekomendacyjne: filtrowanie kolaboracyjne
- Ekstrakcja słów kluczowych
- Znajdowanie definicji
- Analiza opinii
- Rankowanie produktów
- Systemy antyspamowe
- ...

Sposoby tworzenia rankingu

- Stopień istotności
  - binarne: istotne vs nieistotne
  - wiele uporządkowanych kategorii: idealne $>$ doskonałe $>$ dobre $>$ akceptowalne $>$ złe
- Preferencja w parach
  - dokument A lepiej pasuje niż dokument B do zapytania
- Całkowity porządek (posortowanie)
  - dokumenty są posortowane $\{A, B, C, \dots\}$ według ich dopasowania (istotności, relewancji)

Ogólny proces „Learning to rank”

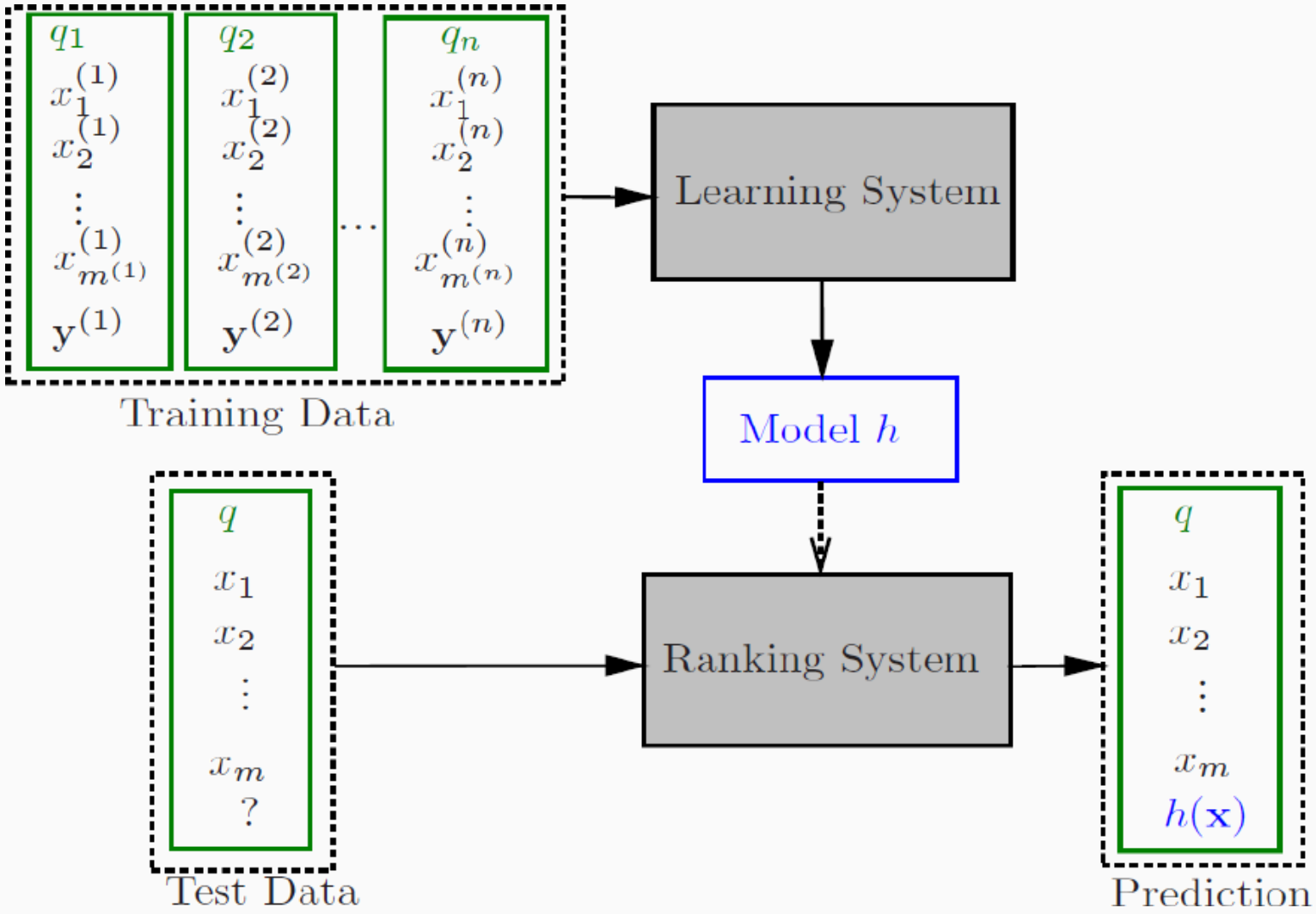

Figure 2: Źródło: MM. Chaa, O. Nouali and K. Bal, “Learning to rank in XML information retrieval: Which feature improve the best?”

Cechy wykorzystywane w procesie uczenia

- Oparte o zestaw cech *(feature)*
  - dokumenty reprezentowane przez wektor cech,
  - możliwość łączenia dużej ilości cech.
- Trenowanie dyskryminacyjne
  - automatyczny proces uczenia oparty na danych treningowych,
  - duże wymaganie dla rzeczywistych wyszukiwarek internetowych.

Kategorie cech

Podział cech / czynników / sygnałów rankujących na kategorie

- Niezależne od zapytania, statyczne
  - cechy które zależą tylko od dokumentu, ale nie od zapytania
  - przykład: PageRank, długość dokumentu
- Zależne od zapytania, dynamiczne
  - zależą od dokumentu i od zapytania
  - przykład: TF-IDF (ważenie względną częstościa termów)
- Zależne tylko od zapytania
  - przykład: ilość słów w zapytaniu

Oceny rankowania

- Oparte o zapytanie: każde zapytanie daje ten sam wkład do miary oceny
  - obliczone na dokumentach powiązanych z tym samym zapytaniem
  - ograniczony zakres wartości miary dla każdego zapytania
  - uśrednione po wszystkich zapytaniach w zbiorze testowym
- Zależne od pozycji: jawne użycie kolejności elementów w rankingu
  - obiekty na wyższych pozycjach są bardziej ważne
  - porządek względny vs. ocena istotności dla każdego dokumentu
  - często nieciągłe i nieróżniczkowalne zwn. oceny istotności

Przykłady miar ocen rankingu

- Mean Average Precision (MAP)
- (Normalized) Discounted Cumulative Gain (DCG i NDCG)
- Precision@n oraz NDCG@n
- Mean Reciprocal Rank (MRR)
- Kendall's tau
- Spearman's rho
- Expected Reciprocal Rank (ERR)
- Yandex's pfound

Podział algorytmów - learning to rank

- Podejście punktowe
  - regresja
  - klasyfikacja
  - regresja porządkowa *(ordinal regression)*
- Podejście par (porównania par)
  - zwykle minimalizacja średniej ilości inwersji
  - można się ograniczyć do par relewantny–nierelewantny
- Podejście list
  - minimalizacja funkcji strat
  - bezpośrednia optymalizacja miary oceny

Algorytmy Learning-to-rank

Podejście punktowe

- przestrzeń wejść
  - pojedynczy dokument y_j
- przestrzeń wyjść
  - regresja: liczba rzeczywista
  - klasyfikacja: nieuporządkowane kategorie
  - regresja porządkowa: uporządkowane kategorie
- przestrzeń hipotez
  - funkcja oceny $f(x)$
- funkcja strat
  - $L(f; x_i, y_j)$

Podejście punktowe - problemy

- Właściwości oceny jakości wydobycia informacji nie są dobrze uwzględnione
- Jest ignorowany fakt, że niektóre dokumenty powiązane są z tym samym zapytaniem, a niektóre nie.
    - Kiedy liczba istotnych dokumentów zmienia się znacząco od zapytania do zapytania, całkowita funkcja strat będzie zdominowana przez zapytania z dużą ilością dokumentów.
- Pozycja dokumentu w rankingu jest niewidoczna dla funkcji strat.
    - Oznacza to, że punktowa funkcja strat może niejawnie podkreślić zbytnio znaczenie tych nieistotnych dokumentów.

Podejście punktowe - inne przykłady

- OPFR, regresja wielomianowa (1989)
- SLR: Staged Logistic Regression (1992)
- Regression Tree for Ordinal Class Prediction (2000),
- Pranking – regresja porządkowa (2002)
- Ranking with Large Margin Principles (2002)
- COR: Constraint Ordinal Regression (2005)
- McRank (2007)
- CRR: Combined Regression and Ranking (2010) kombinacja podejścia punktowego i porównania par

Podejście parami

- przestrzeń wejść
  - pary dokumentów (x_u, x_v)
- przestrzeń wyjść
  - preferencja jednego dokumentu względem drugiego: $y_{u,v} \in \{-1, +1\}$
- przestrzeń hipotez
  - funkcja preferencji $h(x_u, x_v) = 2I_{\{f(x_u) > f(x_v)\}} - 1$
- funkcja strat
  - funkcja strat klasyfikacji par $L(h; x_u, x_v, y_{u,v})$

Podejście parami - rozszerzenia

- Kiedy konstruuje się pary dokumentów, tracona jest informacja o kategorii (pozycji w rankingu): różne pary etykiet traktowane są identycznie
  - czy można wykorzystać więcej informacji o kategoriach uporządkowanych?
  - czy można użyć innego mechanizmu uczącego dla różnych rodzajów par?
- Rozwiązanie:
  - Ranker multi–hiperpłaszczyznowy
  - K kategorii, $K(K-1)/2$ par kategorii
  - agregacja rank $f(x) = \sum_{k,l} \alpha_{k,l} f_{k,l}(x)$
  - problem: nierówna dystrybucja liczb par dokumentów na zapytanie

Podejście parami - inne przykłady

- MART: Multiple Additive Regression Trees (1999)
- RankSVM (2000)
- RankBoost (2002)
- RankNet (2005)
- IR-SVM (2006)
- Frank / FRank (2007)
- RankRLS (2007)
- SortNet (2008)
- MPBoost (2009)
- CRR: Combined Regression and Ranking (2010) kombinacja podejścia punktowego i porównania par
- LambdaRank (2006) kombinacja podejścia porównania par i listowego
- LambdaSMART / LambdaMART (2008) kombinacja podejścia porównania par i listowego

Podejście listowe

- przestrzeń wejść
  - zbiór dokumentów $\mathbf{x} = \{x_j\}_{j=1}^{m}$
- przestrzeń wyjść
  - minimalizacja funkcji strat: permutacja π_y
  - bezpośrednia optymalizacja funkcji oceny: uporządkowane kategorie $\mathbf{y} = \{y_j\}_{j=1}^{m}$
- przestrzeń hipotez
  - minimalizacja funkcji strat: $h(\mathbf{x}) = \text{sort} \circ f(\mathbf{x})$
  - bezpośrednia optymalizacja funkcji oceny: $h(\mathbf{x}) = f(\mathbf{x})$
- funkcja strat
  - $L(h; \mathbf{x}, \pi_y)$ lub $L(h; \mathbf{x}, \mathbf{y})$

Podejścia listowe - dwie główne gałęzie

- Bezpośrednia optymalizacja miary jakości wydobycia informacji
  - Próba optymalizacja funkcji oceny rankowania, albo przynajmniej czegoś związanego z funkcją oceny (wygładzenie, regularyzacja)
- Minimalizacja funkcji strat
  - Funkcja strat zdefiniowana na permutacjach, zaprojektowana na podstawie właściwości procesu tworzenia rankingu

Problemy podejścia bezpośredniej optymalizacji

- Problem
  - Funkcje oceny często nieciągłe i nieróżniczkowalne, ponieważ zależą od pozycji dokumentów w rankingu
  - Typowe metody optymalizacji zakładają ciągłość, a często różniczkowalność
- Próby rozwiązania
  - Przybliżenie funkcji oceny (np. przez jej wygładzenie)
  - Znalezienie gładkiego i różniczkowalnego ograniczenia od góry na funkcję oceny
  - Bezpośrednia optymalizacja, np. algorytmy genetyczne

Problemy podejścia listowej funkcji strat

- Idea: zdefiniowanie funkcji strat opartej na rozumieniu szczególnych właściwości procesu rankowania
- Problemy
  - która funkcja strat jest bliższa prawdy?
  - reprezentacja rankingu jako rozkładu prawdopodobieństwa permutacji prowadzi do dużej złożoności

Przykładowe algorytmy podejścia listowego

- AdaRank (2007)
- ListNet (2007)
- RankCosine (2007)
- SVM^{map} (2007)
- GPRank / RankGP (2007)
- SoftRank (2008)
- NCGD-Boost (2010)
- ES-Rank (2017)
- LambdaRank (2006)

  kombinacja podejścia porównania par i listowego
- LambdaSMART / LambdaMART (2008)

  kombinacja podejścia porównania par i listowego

Porównanie różnych podejść

- Punktowe:
  - modelowanie: regresja / klasyfikacja / regresja porządkowa,
  - zalety: łatwo wykorzystać istniejące teorie i algorytmy,
  - wady: dokładny ranking $\neq$ dokładna punktacja lub kategoria.
- Parami:
  - modelowanie: klasyfikator dla par,
  - wady: informacja o położeniu niewidoczna dla funkcji strat.
- Listowe:
  - modelowanie: ranking,
  - zalety: oparte o zapytanie i o pozycję,
  - wady: bardziej złożone, wymaga nowej teorii.

Rozważany problem

Opis problemu

- Dla danych treningowych które są **binarne** (tak/nie - powiązane z zapytaniem lub nie),

lub

- dla danych częściowo oznaczonych **tylko pozytywną** klasą (semi-supervised),

zbuduj model który jest **bardziej szczegółowy** (wszystkie dokumenty są posortowane względem ich istotności).

Przykład 1

Szukanie błędów w kodzie źródłowym na podstawie raportów błędów

- Dla danego raportu błędu - znajdź pliki kodu źródłowego gdzie wymagana jest interwencja.
  - Zbiór treningowy/testowy:
    - pliki relewantne - kilka/kilkanaście plików w których były naprawy,
    - pliki nierelewantne dla raportu - pozostałe pliki.
    - **Uwaga** Wśród plików nierelewantnych mogą być pliki blisko powiązane z rozpatrywanym błędem jak i zupełnie nie powiązane.
  - Zbiór treningowy jest **binarny** ale problem jest bardziej szczegółowy,
    - zarówno wśród plików relewantnych i nierelewantnych jest pewna hierarchia.

Przykład 2

Przygotowanie przeglądu literatury

- Dla zbioru publikacji tematycznie powiązanych (np. wybranych za pomocą słów kluczowych) znajdź porządek względem podobieństwa do wybranego podzbioru publikacji (zawężającego temat)
  - Zbiór treningowy:
    - Zbiór pozytywnych przykładów (podzbiór zawężający temat),
    - Pozostałe publikacje - nieoznaczone.
  - W zbiorze treningowym mamy tylko kilka **pozytywnych** przykładów i wiele nieoznaczonych.
- Wariant semi-supervised.

Nowa metoda uczenia do rankowania

Ogólny schemat algorytmu

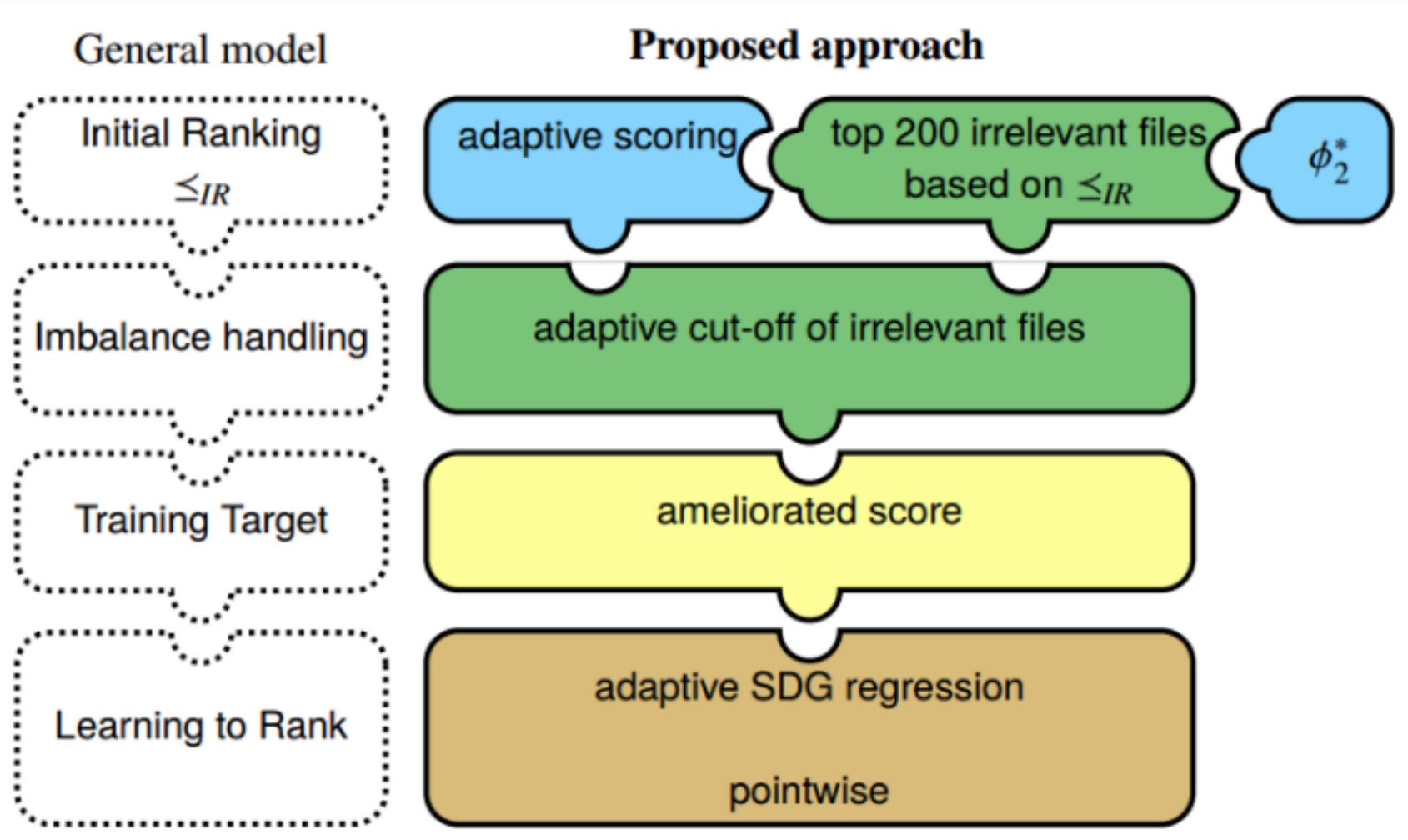

Figure 3: Algorytm

Algorytm

- Naszym celem jest skonstruowanie funkcji rankingowej

$$r : Q \times D \to R.$$

  - Ta funkcja dla zapytania q i dokumentu d określa, w jaki sposób ten dokument jest powiązany z zapytaniem.
  - Para $(q, d) \in Q \times D$ może być przedstawiona jako zbiór cech $[\phi_i(q, d)]$.
  - Funkcja rankingu musi być praktycznie obliczalna nawet w przypadku dużych ilości dokumentów.
  - Proces musi być adaptacyjny - nie oparty na ręcznym doborze parametrów.
- Funkcja rankingu jest tworzona z wykorzystaniem algorytmu uczenia się rankowania.
  - Aby nauczyć algorytm potrzebny jest zbiór treningowy.

Algorytm - wstępny ranking

- W celu przygotowania zbioru treningowego używamy wstępnego rankingu tworzącego porządek dla każdej pary (q, d)
  - Wartość to suma wag otrzymanych z funkcji $p_i(q, d) = \sum_k w_k \cdot \phi_k(q, d)$.
  - Wagi są przygotowywane przez jedną z 11 funkcji statystycznych.
  - Motywacją było wykorzystania testów statystycznych przy wyborze cech.
    - Potencjalnym problemem jest zastosowanie wyników testów do stworzenia rankingu czego nie uzasadniliśmy teoretycznie.
- Najlepiej dopasowana funkcja pod względem metryki MAP jest używana do przygotowania celu treningowego.

Algorytm - budowa wstępnego rankingu

Based on	Weights
W statistics from Levene test for equality of variances with median as center function [82]	$w_i = W_{\phi_i} / \sum_{j=1}^{n} W_{\phi_j}$
H statistics from Kruskal-Wallis H-test [73]	$w_i = H_{\phi_i} / \sum_{j=1}^{n} H_{\phi_j}$
T statistics from T-test for independent samples [107]	$w_i = T_{\phi_i} / \sum_{j=1}^{n} T_{\phi_j}$
χ^2 statistics from chi-square test [107]	$w_i = \chi_{\phi_i} / \sum_{j=1}^{n} \chi_{\phi_j}$
Features weights computed by AdaBoost SAMME.R Classifier [38, 49]	model specific
Features weights computed by Extremely randomized trees classifier[40]	model specific
Features weights computed by Gradient Boosting regression [50]	model specific
Mutual Information between Discrete and Continuous Data Sets [125] denoted as I	$w_i = I(\phi_i, Y) / \sum_{j=1}^{n} I(\phi_j, Y)$
Index of dispersion [107]	$w_i = {}^{D_{\phi_i^{fix}}}/_{D_{\phi_i^{irr}}}$ where $D_{\phi_i} = \sigma^2_{\phi_i}/\mu_{\phi_i}$
Maximum absolute deviation variances [107]	$w_i = \text{mean}\,(\lvert\phi_i^{fix} - \max \phi_i^{fix}\rvert)$
Predefined set of weights	$w_i = 0.5$

Figure 4: Funkcje wag

Algorytm - przygotowanie zbioru treningowego

- Cel treningowy - ulepszony, aby zapewnić wyższą rangę dokumentom związanym z zapytaniem

$$p^*(q,d) = \sum_{i=1}^{n} w_i \cdot \phi_i(q,d) + 1_{[d \in relevant(q)]} \cdot max_{i=1..n} \phi_i(q,d).$$

Algorytm - obsługa balansu

- W celu zapewnienia proporcji między dokumentami powiązanymi i niepowiązanymi do zbioru treningowego trafiają:
  - wszystkie dokumenty powiązane z zapytaniem q
  - fragment niepowiązanych dokumentów

$$t_d(q) = relevant(q) \cup \{d \notin relevant(q) \mid 0 < p^*(q,d) \leq c_q(d)\},$$

    - zależny od procentowego udziału $c_q(d)$.

Algorytm - ranking

- Regresja SGD, z adaptacyjnym doborem parametrów spośród następujących:
  - Funkcje strat: Least Squares, Huber, epsilon insensitive, squared epsilon insensitive.
  - Regularyzacja: brak, przez normę L1, normę L2 lub Elastic Net.
- Najlepiej dopasowane parametry są wybierane za pomocą metryki MAP.

Dyskusja

- Budowa wstępnego rankingu - jest zbudowana z testów statystycznych.
  - Inspirujemy się metodami które sortują istotność cech.
- Jak możemy ulepszać cel treningowy?
- Wybór początkowej cechy używanej do zapewnienia balansu pomiędzy dokumentami powiązanymi i pozostałymi.
  - Wybraliśmy eksperymentalnie cechę 2 w zastosowaniu do lokalizacji błędów.
  - Używamy procentowego doboru przykładów negatywnych.
- Metoda obecnie opiera się o grid search w 3 etapach.

Zastosowania: Adaptacyjna lokalizacja błędów

Adaptacyjna lokalizacja błędów

- Problem:
  - Użytkownik zgłasza błąd do systemu zarządzania błędami.
  - Następnie programista musi zlokalizować odpowiednie pliki źródłowe w projekcie i je zmodyfikować.
  - Istnieje różnica między językiem naturalnym używanym w raportach błędów a językiem programowania używanym w kodzie źródłowym.
- Motywacja: Lokalizację błędów można (częściowo) zautomatyzować, analizując repozytorium projektu i wybierając pliki podobne do zgłoszeń błędów.
  - Lokalizacja błędu skraca czas poświęcony na przygotowanie poprawki.

Adaptacyjna lokalizacja błędów

- Budowa funkcji rankingowej $r : B \times S \to R$.
  - Ta funkcja dla raportu o błędzie b i pliku s określa, w jaki sposób ten plik jest powiązany z raportem o błędzie.
- Używamy tego samego zestawu 19 cech zaproponowanego przez Ye i innych.
  - Oprócz cechy 2, którą poprawiliśmy, orginalna była błędnie przygotowana.
- Historia projektu jest podzielona na kolejne grupy o wielkości 500 raportów.
  - Jest to ten sam rozmiar jak u Ye i innych.
  - Cel treningowy jest przygotowywany od grupy n do rankowania w grupie n + 1.

Adaptacyjna lokalizacja błędów - porównanie algorytmów

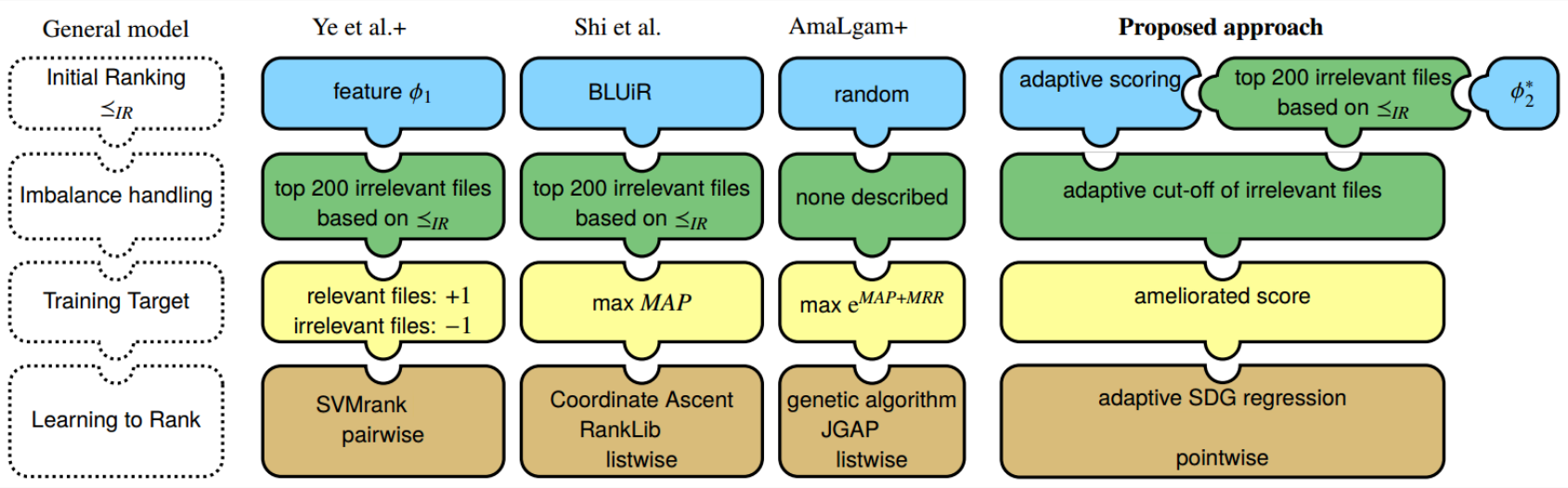

Figure 5: Algorytmy

Adaptacyjna lokalizacja błędów - eksperymenty

- Zestawy danych:
  - Ye i inni - używany do porównania z najnowszą metodą.
    - Składa się z 6 projektów (AspectJ, Birt, Eclipse UI, JDT, SWT, Tomcat).
    - Wszystkie projekty mają brakujące opisy błędów, dlatego sprawdziliśmy nasz algorytm na dwóch wersjach zbioru danych - z brakującymi i dodanymi opisami.
  - Zbiór danych BugLocator
    - Powszechnie używany w wielu publikacjach, starszy niż najnowocześniejszy zbiór danych Ye i innych.
    - Użyto projektu Eclipse 3.1, ponieważ pozostałe mają zbyt mało raportów, aby stworzyć zestaw treningowy.
- Metryki:
  - Accuracy@k, $k \in 1, 2, ..., 20$
  - MAP
  - MRR

Adaptacyjna lokalizacja błędów - wyniki

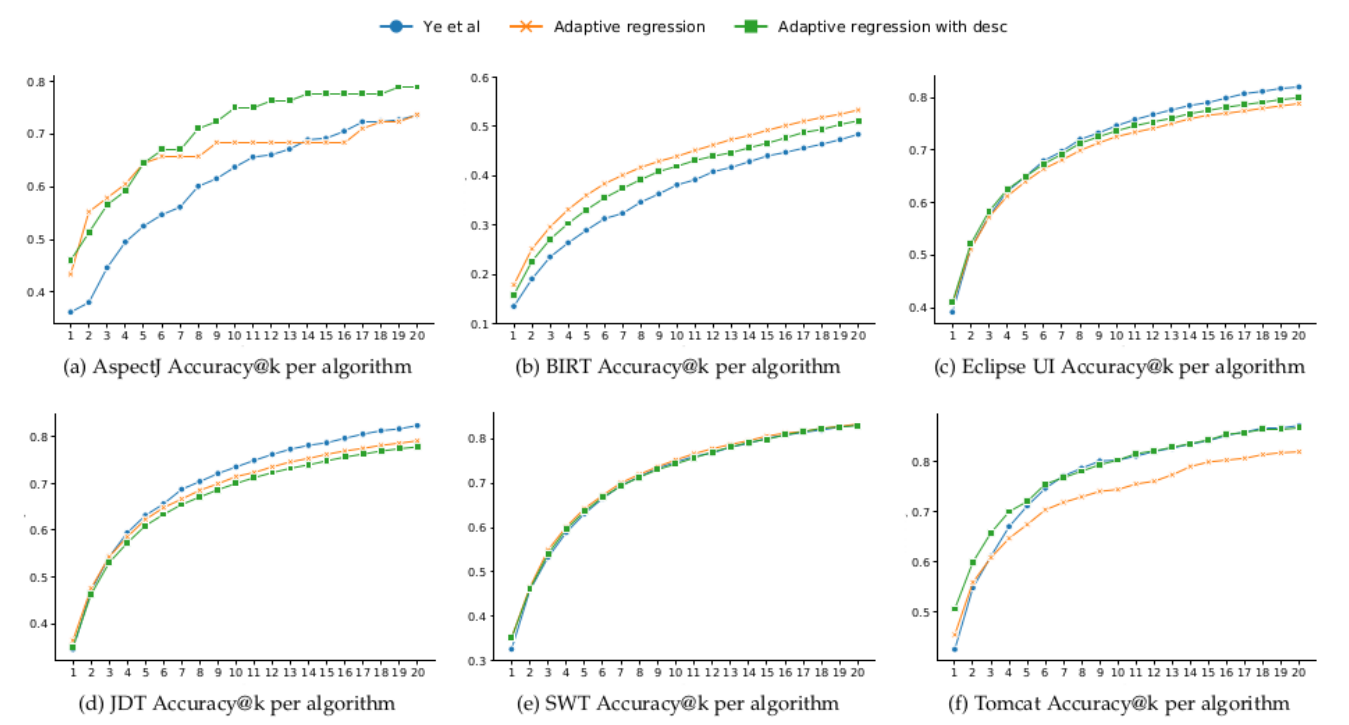

Figure 6: Accuracy@k

Adaptacyjna lokalizacja błędów - wyniki

	MAP						MRR					
Method	AspectJ	BIRT	Eclipse	JDT	SWT	Tomcat	AspectJ	BIRT	Eclipse	JDT	SWT	Tomcat
Adaptive regression	0.45	0.21	0.44	0.40	0.42	0.50	0.53	0.27	0.52	0.48	0.48	0.56
Adaptive regression with desc	0.46	0.19	0.45	0.39	0.41	0.54	0.54	0.25	0.52	0.47	0.48	0.61
Ye et al.+	0.37	0.16	0.44	0.39	0.40	0.49	0.44	0.21	0.51	0.47	0.46	0.55

Figure 7: MAP i MRR

Adaptacyjna lokalizacja błędów - wyniki (BugLocator/Eclipse 3.1)

Method	Acc@1	Acc@5	Acc@10	MAP	MRR
BugLocator	0.291	0.538	0.626	0.3	0.41
BRTTracer	0.326	0.559	0.652	0.33	0.43
BLUiR	0.329	0.562	0.654	0.33	0.44
Ye et al.+	0.34	0.57	0.66	0.34	0.45
AmaLgam+	0.357	0.603	0.691	0.36	0.47
ConCodeSe	0.376	0.612	0.699	0.37	0.57
Shi et al.	0.297	0.664	0.85	0.306	0.399
Adaptive regression	0.7	0.752	0.785	0.6	0.728

Figure 8: Accuracy@k, MAP i MRR

Adaptacyjna lokalizacja błędów - dyskusja

- Algorytm jest pierwszym udanym użyciem podejścia punktowego w lokalizacji błędów:
  - Punktowa konstrukcja celu treningowego pozwala poświęcić więcej czasu na wybór najlepszych parametrów.
  - Nasz model ma parametry wybrane i dostosowane automatycznie do projektu / zbioru danych.

Potencjalne zagrożenia

- Brak opisów raportów o błędach - błąd zbiorze danych Ye i innych, w niektórych raportach brakuje danych:
  - Wykonaliśmy eksperymenty z obiema wersjami zbioru danych.
- Jednorodność projektów:
  - Wszystkie oceniane projekty są projektami Java typu open source o podobnym poziomie jakości kodu źródłowego.

Adaptacyjna lokalizacja błędów - wnioski

- Proponujemy nowe, adaptacyjne podejście do problemu:
  - Adaptacja działa bez konieczności wykonywania oddzielnych procedur dopasowywania parametrów.
  - Algorytm będzie z czasem dostosowywał się do zmian w charakterystyce projektu.
- Poprawiamy metryki Accuracy @ k, k = 1, MAP i MRR we wszystkich ocenianych projektach dla obu zbiorów danych.
- Proponowany algorytm ma podobną złożoność czasową, jak Ye i innych.
- Nasza metoda wykorzystuje więcej kroków (wstępny ranking, obsługa niezbalanosowanych klas).

Zastosowania: Wybór publikacji powiązanych

Wybór publikacji powiązanych z dziedziną mikrobiomu ludzkiego.

- Problem:
  - Badacz przygotowuje listę publikacji używających uczenia maszynowego w dziedzinie mikrobiomu ludzkiego.
  - Tylko część publikacji jest poprawnie oznaczona.
  - Istnieje duża ilość publikacji potencjalnie powiązanych, wymagających sprawdzenia.
- Motywacja: Sprawdzenie powiązań można zautomatyzować.
  - Znalezienie powiązanych publikacji skraca czas poświęcony na badania.
  - Wcześniej nieznane publikacje zostają przypisane do wybranej dziedziny.

Wybór powiązanych publikacji - zbieranie danych

- Badacze publikują materiały pomocnicze, takie jak kod źródłowy i dane, obok artykułu.
- Dostęp do kodu źródłowego wspierającego badania pozwala na szybszą replikację i walidację wyników.
- Aby przygotować wstępny zbiór publikacji repozytoria portalu GitHub zostały przeszukane pod względem występowania linków do publikacji.
    - Użyte zostały tagi “ML” oraz “HUMAN” dla badań mikrobiomu ludzkiego.
- Cechy dla publikacji zostały przygotowane w oparciu o model tf-idf.

Wybór powiązanych publikacji - algorytm

- Wstępny ranking dla dziedziny reprezentowanej przez tag q

$$p_i(q,d) = \sum_k w_k^q * \phi_k(d)$$

- Cel treningowy dla dziedziny reprezentowanej przez tag q

$$p^*(q,d) = \sum_{i=1}^{n} w_i \cdot \phi_i(d) + 1_{[d \in relevant(q)]} \cdot max_{i=1..n} \phi_i(d).$$

- Połączony cel treningowy, dla tagów “ML” oraz “HUMAN”

$$p^*("Combined", d) = 0.5 * p^*(ML, d) + 0.5 * p^*(HUMAN, d).$$

Wybór powiązanych publikacji - eksperymenty

- Zestaw danych - składający się z 410 publikacji:
  - 20 oznaczonych tagami („HUMAN” i / lub „ML”), pozostałe nie oznaczone.
- Po nauczeniu metody pierwsze 20 najlepszych wyników zostało sprawdzonych ręcznie pod względem przypisania do tagów.

Wybór powiązanych publikacji - rezultaty

- Spośród 410 zidentyfikowanych artykułów opartych na wyszukiwaniu słów kluczowych,
  - metoda oznaczyła 29 publikacji jako powiązane z dziedziną mikrobiomu ludzkiego,
  - z których 17 znalazło się na ostatecznej liście po ręcznej weryfikacji.

Dyskusja

Dyskusja

- Budowa wstępnego rankingu - jest zbudowana z testów statystycznych.
  - Inspirujemy się metodami które sortują istotność cech.
- Jak możemy ulepszać cel treningowy?
- Wybór początkowej cechy używanej do zapewnienia balansu pomiędzy dokumentami powiązanymi i pozostałymi.
  - Wybraliśmy eksperymentalnie cechę 2 w zastosowaniu do lokalizacji błędów.
  - Używamy procentowego doboru przykładów negatywnych.
- Metoda obecnie opiera się o grid search w 3 etapach.

Publikacje

- Mikołaj Fejzer, Jakub Narębski, Piotr Przymus and Krzysztof Stencel, “Tracking Buggy Files: New Efficient Adaptive Bug Localization Algorithm”, recently accepted to IEEE Transactions on Software Engineering. doi: 10.1109/TSE.2021.3064447
- Laura Judith Marcos-Zambrano i inni, “Applications of machine learning in human microbiome studies: a review on feature selection, biomarker identification, disease prediction and treatment.” Frontiers in microbiology 12 (2021): 313.

Referencje

- MM. Chaa, O. Nouali i K. Bal, “Learning to rank in XML information retrieval: Which feature improve the best?”, Seventh International Conference on Digital Information Management (ICDIM 2012), 2012, pp. 336-340, doi: 10.1109/ICDIM.2012.6360123.
- X. Ye, R. Bunescu i C. Liu, “Mapping Bug Reports to Relevant Files: A Ranking Model, a Fine-Grained Benchmark, and Feature Evaluation,” IEEE Transactions on Software Engineering, vol. 42, no. 4, pp. 379-402, 1 April 2016, doi: 10.1109/TSE.2015.2479232.
- Tie-Yan Liu, “Learning to Rank for Information Retrieval” https://www.scribd.com/document/41516616/t7a-Learning-to-Rank-Tutorial