Pairwise Learning to Rank Approach with Gradient Tree Boosting Algorithm

Chatrin Pandiangan
Fakultas Informatika dan Teknik Elektro
Institut Teknologi Del
Medan, Indonesia
chatrinpandiangan@gmail.com

Kiky Purnamasari Napitupulu
Fakultas Informatika dan Teknik Elektro
Institut Teknologi Del
Porsea, Indonesia
kikynapitupulu687@gmail.com

Debby Debora Hutajulu
Fakultas Informatika dan Teknik Elektro
Institut Teknologi Del
Medan, Indonesia
debbydbrrh@gmail.com

Fivin Sadesla Tambunan
Fakultas Informatika dan Teknik Elektro
Institut Teknologi Del
Tarutung, Indonesia
fivintambunan@gmail.com

Abstrak—Learning to Rank **adalah bagian dari penerapan** ***machine learning*** **dalam pembangunan model perankingan pada sistem temu balik informasi. Salah satu pendekatan untuk membangun model** ***Learning to Rank*** **yaitu** ***pairwise approach*** **dimana pasangan dokumen dianggap sebagai** ***input*** **untuk sistem pembelajaran. Dalam meningkatkan kinerja model perankingan yang akan dibangun diterapkan algoritma** ***gradient tree boosting*****. Salah satu library yang menerapkan** ***gradient tree boosting*** **terbaik yang saat ini tersedia adalah XGBoost (*eXtreme Gradient Boosting*) yang memuat *package* XGBRanker.**

Kata Kunci—information retrieval, learning to rank, pairwise approach, gradient tree boosting

I. PENDAHULUAN

Information Retrieval (IR) telah menjadi bagian dari perkembangan teknologi manusia yang signifikan sejak dikenalnya penyampaian informasi melalui tulisan. IR ini akan berkaitan dengan penyimpanan, pengorganisasian, dan pencarian kumpulan informasi [1]. Tujuan dari sistem IR adalah untuk memilih item informasional (teks, gambar, video, dan lain sebagainya) atau dapat disebut sebagai dokumen yang diharapkan releven sesuai dengan informasi yang dicari pengguna berdasarkan koleksi item tersebut. Namun, saat ini koleksi tersebut berkisar dari sekumpulan item kecil pada perangkat individu pengguna hingga sumber daya *World Wide Web* yang luas [1].

Dalam semua kasus pencarian, tugas yang perlu dilakukan adalah sama yaitu mengekstrak beberapa set item yang ingin dimiliki oleh pencari dari semua item yang tidak pengguna inginkan [1] dan melakukan pemeringkatan item. Ini bukanlah tugas yang sederhana dan bukan hanya melibatkan aspek teknis membangun sistem untuk melakukan seleksi pencarian tersebut, tetapi juga aspek psikologi dan perilaku pengguna untuk memahami apa yang membedakan item yang diinginkan dari yang tidak diinginkan berdasarkan sudut pandang pengguna tersebut [1]. Salah satu bidang yang dapat menyelesaikan masalah pengambilan informasi atau *information retrieval* (IR) ini adalah *Learning to Rank* (LTR) [2].

Learning to Rank (LTR) merupakan sebuah bidang penelitian yang mengambil bagian dalam pengambilan informasi atau yang disebut dengan *information retrieval* (IR). LTR ini mempelajari fungsi perankingan dengan memberikan bobot pada setiap fitur dokumen dan kemudian menggunakan fungsi perankingan tersebut untuk memperkirakan skor relevansi setiap dokumen, serta memeringkatkan dokumen-dokumen tersebut berdasarkan perkiraan skor relevansi [2]. LTR juga bagian dari algoritma yang menangani pengurutan data [3] dan bagian dari *supervised machine learning* yang digunakan untuk membangun model klasifikasi sistem temu balik informasi [4]. Untuk mengurutkan data ini, perlu diketahui item-item mana yang lebih penting dan perlu ditampilkan terlebih dahulu pada pengguna [3].

Terdapat beberapa pendekatan yang dapat digunakan untuk membangun model LTR yaitu *pointwise approach* yang diterapkan ke setiap dokmen secara individual dan menganggapnya terpisah dari data latih lainnya [3], *pairwise approach* yang menghitung *priority* setiap kemungkinan pasangan kueri dan dokumen lalu mengurutkannya, dan *listwise approach* yang membandingkan skor setiap pasangan kueri dan dokumen pada *list rank* yang telah diperoleh.

Pada beberapa penelitian sebelumnya, permasalahan IR ini dilakukan dengan menggunakan konsep klasifikasi. Dimana, model LTR dibangun dengan menggunakan klasifikasi biner untuk memisahkan dokumen yang relevan dan tidak relevan, serta dengan menghitung probabilitas dokumen yang menjadi relevan untuk pengurutan dokumen [4]. Pendekatan klasifikasi ini bekerja dengan cukup baik tetapi memiliki beberapa batasan yaitu hanya mempertimbangkan satu elemen pada satu waktu tertentu dan menyimpan dokumen lain dalam kondisi isolasi penuh. Dengan kata lain, model LTR dengan pendekatan klasifikasi hanya dapat memutuskan apakah suatu dokumen relevan dengan hanya melihat fitur pada dokumen tertentu dan tidak melihat fitur pada dokumen lain [4].

Dalam penelitian ini, diusulkan pendekatan dengan menggunakan *pairwise approach* untuk membangun model perankingan dimana, pasangan dokumen dianggap sebagai *input* untuk sistem pembelajaran serta menggunakan algoritma *gradient tree boosting* untuk meningkatkan kinerja model dalam kasus *Learning to Rank*.

II. METODOLOGI

Gradient tree boosting adalah sebuah teknik untuk meningkatkan kinerja model *tree* dengan menggunakan prosedur penurunan *gradient* dalam meminimalkan nilai dari *loss function.* Salah satu *library* yang menerapkan *gradient tree boosting* terbaik yang saat ini tersedia adalah XGBoost (eXtreme Gradient Boosting) [5].

A. *Pairwise Approach*

Dalam *pairwise approach*, peringkat diubah menjadi klasifikasi berpasangan atau regresi berpasangan. Di *former case*, pengklasifikasi untuk mengklasifikasikan urutan peringkat pasangan dokumen dibuat dan digunakan di peringkat dokumen. Dalam *pairwise approach*, struktur peringkat grup juga diabaikan [6].

Dalam pendekatan *pairwise*, pasangan dokumen dianggap sebagai *input* untuk sistem pembelajaran. Tujuannya adalah bukan untuk menentukan skor relevansi setiap dokumen dalam kaitannya dengan kueri, tetapi dokumen mana yang lebih relevan daripada yang lain.

Keuntungan dengan menggunakan pendekatan pairwise adalah metodologi yang ada mengenai klasifikasi dapat diterapkan secara langsung, kemudian contoh pelatihan pada pasangan dokumen dapat dengan mudah diperoleh dalam skenario tertentu.

Pendekatan *pairwise* bekerja lebih baik dalam praktiknya daripada pendekatan *pointwise* karena memprediksi urutan relatif lebih dekat dengan sifat peringkat daripada memprediksi label kelas atau skor relevansi. Dalam literatur *learning to rank*, terdapat beberapa algoritma *ranking pairwise* yang diusulkan, berdasarkan boosting (Freund, 2003), *support vector machine* (Joachims, 2002), *neural network* (Burges, 2005) dan mesin pembelajaran lainnya [7].

B. *Gradient Tree Boosting*

Salah satu cara untuk meningkatkan performa model adalah dengan menurunkan nilai penurunan paling rendah dalam *loss function* model yaitu gradien. Gradien atau multi-variabel *derivative* merupakan arah tanjakan paling curam, ini dapat dilihat dengan menghitung *gradient negative* yang menghasilkan arah penurunan yang paling curam. Teknik ini disebut dengan penurunan gradient (*descent gradient*) [8].

Gradient tree boosting atau disebut dengan *gradient boosting* adalah sebuah teknik untuk meningkatkan kinerja model *tree* dengan menggunakan prosedur penurunan *gradient* dalam meminimalkan nilai dari *loss function* [9]. *Gradient boosting* termasuk pada algoritma *greedy* yang dapat menyesuaikan data latih dengan cepat dan mengurangi *overfitting* pada model. salah satu implementasi *gradient tree boosting* terbaik yang tersedia saat ini adalah XGBoost (eXtreme Gradient Boosting) [5].

Modul ansambel Scikit-learn menyediakan kelas GradientBoostingClassifier dan GradientBoostingRegressor untuk peningkatan gradien menggunakan *decision tree*. *Tree* ini akan meningkatkan kinerjanya melalui penurunan gradient [8].

$\widetilde{y_{i,k}} = 1, if\ y_i = k, and\ \widetilde{y_{i,k}} = 0\ otherwise.$
$F_{i,k} = 0, k = 0\ to\ K-1 = 1\ to\ N$
For $m = 1\ to\ M\ do$
  For $k = 0\ to\ K-1\ Do$
    $p_{i,k} = \exp(F_{i,k}) / \sum_{s=0}^{K-1} eexp(F_{i,s})$

$\{R_{j,k,m}\}_{j=1}^{J} = J - terminal\ node\ regression\ tree\ for$
    $\{\widetilde{y_{i,k}} - p_{i,k}, X_i\}_{i=1}^{N}$
    $\beta_{j,k,m} = \frac{K-1}{K} \frac{\sum_{X_i \in R_{j,k,m}} \widetilde{y_{i,k}} - p_{i,k}}{\sum_{X_i \in R_{j,k,m}} (1 - p_{i,k})\, p_{i,k}}$
  $F_{i,k} = F_{i,k} + v \sum_{j=1}^{J} \beta_{j,k,m} 1_{X_i \in R_{j,k,m}}$
  End
End

Gambar 1. Algoritma *Gradient Boosting Multiple Classification*

Pada Gambar 1 menunjukkan algoritma *gradient boosting* pada kasus *multiple classification* dimana terdapat tiga parameter utama saat mengimplementasikan *gradient boosting* untuk *multiple classification* yaitu sebagai berikut [10]:

- M merupakan jumlah total iterasi *boosting*
- J merupakan ukuran *tree* atau jumlah *terminal nodes*
- v adalah *shrinkage coefficient*

$\widetilde{y_i} = 2^{y_i} - 1$
$S_i = \frac{1}{N} \sum_{S=1}^{N} \widetilde{y_S}\ , i = 1\ to\ N$
For $m = 1\ to\ M\ do$
    $\{R_{j,k,m}\}_{j=1}^{J} = J - terminal\ node\ regression\ tree\ for$
    $\{\widetilde{y_i} - S_i\ ,\ X_i\}_{i=1}^{N}$
    $\beta_{j,k,m} = \sum_{X_i \in R_{j,k,m}} \widetilde{y_i} - S_i$
  $S_i = S_i + v \sum_{j=1}^{J} \beta_{j,k,m} 1_{X_i \in R_{j,k,m}}$
End

Gambar 2. Algoritma *Gradient Boosting Regressions*

Gambar 2 menunjukkan algoritma *gradient boosting* pada kasus *regression* yang mengimplementasikan *least square boosting tree algorithm.* Dimana setelah nilai dari S_i dihitung. Maka dapat secara langsung digunakan sebagai skor peringkat untuk mengurutkan poin data dalam setiap kueri [10].

C. XGBoost

XGboost merupakan *library* pembelajaran mesin yang dominan digunakan untuk kasus klasifikasi dan regresi. Pada prinsipnya, *gradient tree boosting* dibangun secara

berurutan dan secara perlahan berdasarkan data yang dimiliki. Namun, untuk meningkatkan prediksi di setiap iterasi, XGBoost membangun *tree* secara parallel [5]. XGBoost dapat menghasilkan kinerja prediksi yang baik dengan mengontrol kompleksitas model dan mengurangi *overfitting* melalui regularisasi model yang dibangun.

Dalam konsep pencarian informasi dengan *Learning to Rank*, XGBoost dapat digunakan untuk melakukan operasi pengurutan informasi serta mengoptimalkan pengurutan operasi pengurutan tersebut [5]. XGboost akan menyimpan data yang diurutkan dalam unit blok memori kemudian mengurutkan blok tersebut secara efisien dengan menggunakan *parallel* CPU *cores*. Dengan XGBoost, penanganan data berbobot akan dapat dilakukan secara efisien melalui algoritma *quantile sketch*. Algoritma ini dapat secara efisien menangani *sparse data* dan mendukung komputasi *out-of-code* pada kumpulan data besar [5].

Beberapa hal yang perlu dilakukan ketika ingin menerapkan XGBoost pada pencarian informasi dengan konsep LTR adalah melihat posisi dokumen dan hubungannya satu sama lain. Kemudian, untuk setiap kueri, bentuk sekelompok kueri yang dapat dijadikan sebagai kueri khusus dalam mengoptimalkan peringkat dalam semua kelompok kueri tersebut [3]. LambdaMART adalah model yang menggunakan ide ini. Model ini akan melihat pasangan dokumen dan mempertimbangkan urutan relatif dokumen dalam pasangan [3]. Jika urutannya salah (peringkat dokumen yang tidak relevan lebih tinggi dari yang relevan), maka model akan memberikan *penalty*. Sehingga, selama pelatihan model, diharapkan banyak *penalty* yang dihasilkan sekecil mungkin.

XGBoost memuat *packages* XGBranker dimana objektif yang digunakan seperti $rank:pairwise$ dan ukuran evaluasi seperti nilai ndcg (*normalized discounted cumulative gain*) atau $ndcg@n$ (NDCG n elemen pertama pada *list*). Penggunaan $rank:pairwise$ sebagai objektif dikarenakan penelitian ini akan membahas masalah pemeringkatan.

Daftar hasil pencarian bervariasi panjangnya tergantung pada *query*. Membandingkan kinerja *search engine* dari satu *query* ke *query* berikutnya tidak dapat dicapai secara konsisten hanya dengan menggunakan DCG, sehingga perolehan kumulatif di setiap posisi untuk nilai yang dipilih dari p harus dinormalisasi di seluruh kueri. Ini dilakukan dengan menyortir semua dokumen yang relevan dalam korpus menurut relevansi relatifnya, menghasilkan DCG semaksimal mungkin melalui posisi p, juga disebut DCG Ideal (IDCG) melalui posisi itu. Untuk kueri, keuntungan kumulatif diskon yang dinormalisasi, atau NDCG, dihitung sebagai:

$$nDCG_p = \frac{DCG_p}{IDCG_p} \quad (1)$$

Dimana IDCG adalah keuntungan kumulatif diskon yang ideal.

$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{log_2(i+1)} \quad (2)$$

Dan $RELp$ merupakan daftar dokumen yang relevan (diurutkan menurut relevansinya) dalam korpus sampai dengan posisi p.

III. EKSPERIMEN

A. LETOR *Benchmark Datasets*

LETOR adalah kumpulan *dataset* tolak ukur penelitian tentang *Learning to Rank* untuk menentukan peringkat pencarian informasi, yang dirilis oleh Microsoft Research Asia [11].

LETOR berisi fitur standar, penilaian relevansi, partisi data, alat evaluasi, dan beberapa *baseline* lainnya. LETOR dibangun berdasarkan beberapa korpora data dan kumpulan kueri yang telah banyak digunakan di kasus IR. Dokumen dalam korpora akan diambil sampelnya untuk merancang strategi dan kemudian fitur serta metadata yang diektraksi dari setiap pasangan kueri-dokumen [11].

LETOR V 1.0 dirilis pada April 2007. Versi 2.0 dirilis pada Desember 2007. Versi 3.0 dirilis pada Desember 2008. Versi ini, 4.0, dirilis pada Juli 2009. Berbeda dari versi sebelumnya (V3.0 adalah pembaruan berbasis di V2.0 dan V2.0 adalah pembaruan berdasarkan V1.0), LETOR 4.0 adalah rilis yang masih paling baru.

Pada penelitian ini, dataset yang digunakan adalah dataset LETOR 4.0 MQ2008. Dataset dipartisi menjadi 5 *fold* dan terdiri dari 15.211 *instances* pelatihan, dimana masing-masing *instance* pelatihan diwakili dengan 46 fitur sebagai vektor dan diberi label sebagai salah satu penilaian yang relevan seperti 0,1, dan 2.

B. Metrik Evaluasi

Untuk evaluasi performa perankingan *Learning to Rank* dapat dengan menggunakan metrik evaluasi yang mendukung penilaian bertingkat dan memberikan *penalize error* pada awal daftar peringkat yaitu *Discounted Cumulative Gain* (DCG) [12].

$$DCG_n = \sum_{i=1}^{n} \frac{G_i}{log_2(i+1)} \quad (3)$$

Dimana i menunjukkan posisi pada daftar dokumen dan $G_i$menunjukkan fungsi tingkat relevansi. Namun dikarenakan rentang nilai pada DCG tidak konsisten sesuai dengan kueri [12]. Maka, penelitian ini mengadopsi NDCG (*Normalized Discounted Cumulative Gain*) sebagai metrik peringkatan utama dalam melihat kualitas perankingan.

$$NDCG_n = Z_n \sum_{i=1}^{n} \frac{G_i}{log_2(i+1)} \quad (4)$$

Dimana, Z_n menunjukkan faktor normalisasi yang dapat digunakan untuk membuat NDCG *perfect ranking* menjadi bernilai 1 dengan rentang nilai antara 0 sampai 1.

C. Pengaturan Eksperimen

Pada implementasi yang dilakukan dalam penelitian ini, terdapat beberapa *library* yang digunakan, salah satunya adalah *xgboost*. *Library* ini merupakan *library* peningkatan gradien terdistribusi yang dioptimalkan dan dirancang untuk menjadi sangat efisien, fleksibel, dan *portable*. *Library* tersebut mengimplementasikan algoritma pembelajaran mesin di bawah kerangka *Gradient Boosting*. XGBoost menyediakan *parallel tree boosting* (juga dikenal sebagai GBDT, GBM) yang memecahkan banyak masalah ilmu data dengan cara yang cepat dan akurat.

```
import xgboost as xgb
```

Gambar 3. Library Xgboost

LETOR 4.0 MQ2008 *dataset* akan dimuat dalam format *svmlight*. Format ini adalah format berbasis teks, dengan satu sampel per baris. Hal ini tidak menyimpan fitur bernilai nol sehingga cocok untuk dataset yang sifatnya *parse*. Elemen pertama dari setiap baris dapat digunakan untuk menyimpan variabel target untuk diprediksi. Format ini digunakan sebagai format default untuk program baris perintah svmlight dan libsvm.

```
from sklearn.datasets import
load_svmlight_file
```

Gambar 4. Library Scikit Learn

Dataset dipartisi menjadi 5 *fold*, dimana setiap *train set*, *valid test*, dan *test* akan ditransformasikan ke dalam bentuk kelompok-kelompok data dimana dokumen akan dibagi menjadi kelompok kueri.

```
python /content/drive/MyDrive/xgboost-
master/demo/rank/trans_data.py train.txt
mq2008.train mq2008.train.group

python /content/drive/MyDrive/xgboost-
master/demo/rank/trans_data.py test.txt
mq2008.test mq2008.test.group

python /content/drive/MyDrive/xgboost-
master/demo/rank/trans_data.py vali.txt
mq2008.vali mq2008.vali.group
```

Gambar 5. Transformasi Data

Untuk membangun model perankingan dibutuhkan juga XGBRanker dimana parameter yang digunakan dalam penelitian ini ditunjukkan pada TABEL 1.

TABEL 1. PENGATURAN PARAMETER

Parameter	Nilai
Objectives	Rank: Pairwise
Evaluation Metric	NDCG
Learning Rate	0.1
Gamma	1.0
Minimum Child Weight	0.1
Maximum Depth	6
N Estimators	4

Dalam penelitian ini, pemodelan akan dilakukan sebanyak 5 kali dikarenakan LETOR 4.0 MQ2008 memiliki 5 Fold partisi *dataset*. Selain itu, pengukuran perangkingan akan dievaluasi dengan menggunakan *Normalized Discounted Cumulative Gain* (NDCG) sehingga akan terdapat 4 skor NDCG di setiap *fold* sesuai dengan jumlah *estimator* yang sudah didefinisikan.

D. Hasil

Dalam implementasi *Gradient Tree Boosting* pada perankingan, algoritma ini membutuhkan objek-objek yang berada dalam kelompok objek yang sama dan berurutan. Oleh karena itu, *dataset* LETOR 4.0 MQ2008 ditransformasi ke dalam beberapa kelompok dimana setiap kueri pada *train set*, *valid set*, dan *test set* dikelompokkan dan disesuaikan berdasarkan ukuran *array* dari kelompok *dataset* tersebut. Kelompok data ini dibagi menjadi 3 set data yaitu *train set group*, *valid set group*, dan *train set group*.

Train set dan *valid set* akan digunakan untuk membagun model perankingan. Model ini juga dibangun dengan memanfaatkan *library* XGBoost dan XGBRanker beserta parameter pada TABEL 1. Sedangkan *test set* dari setiap *fold* digunakan untuk memprediksi skor perankingan yang dihasilkan model, yaitu sebagai berikut:

- Fold 1: 0.783, 0.168, 0.711, ..., 0.493, 0.673, 0.289
- Fold 2: 0.383, 0.280, 0.495, ..., 0.520, 0.208, 0.669
- Fold 3: 0.445, 0.218, 0.394, ..., 0.181, 0.390 , 0.465
- Fold 4: 0.276, 0.527, 0.153, ..., 0.313, 0.153, 0.193
- Fold 5: 0.608, 0.174, 0.649 , ..., 0.179, 0.383

Selanjutnya, model perangkian dievaluasi dengan metrik evaluasi NDCG. Skor NDCG hasil perangkingan dapat dilihat pada TABEL 2.

TABEL 2. HASIL EVALUASI NDCG

Fold	Estimator 0	Estimator 1	Estimator 2	Estimator 3
1	0.793475	0.79652	0.808798	0.807923
2	0.821042	0.826573	0.831727	0.832029
3	0.808025	0.806311	0.816868	0.819315
4	0.777406	0.791057	0.790832	0.790213
5	0.790501	0.798573	0.804969	0.809962

E. Pembahasan

Berdasarkan skor prediksi perankingan yang diperoleh, skor tersebut hanya berlaku untuk *input* data yang telah dikelompokkan sebelumnya. Hasil evaluasi dari penelitian ini adalah nilai NDCG untuk setiap *fold* seperti yang ditunjukkan pada TABEL 2.

TABEL 3. RATA – RATA HASIL EVALUASI NDCG

Fold	Rata – Rata 4 Estimator	NDCG 5 Fold
1	0.801679	0.806106
2	0.827843	
3	0.812629	
4	0.787377	
5	0.801001	

Pada TABEL 3 dapat dilihat skor NDCG kelima *fold* atau rata-rata skor NDCG setiap *fold* yaitu 0.806106 dari 1 (*perfect ranking*). Skor NDCG ini membuktikan bahwa penerapan algoritma *XGBranker* dan pendekatan *pairwise* sebagai pendekatan untuk membangun model *Learning to Rank*, menghasilkan skor NDCG yang baik untuk setiap 5 Fold partisi *dataset* LETOR 4.0 MQ2008 dan kualitas perankingan yang dihasilkan oleh model juga sudah baik.

IV. KESIMPULAN

Penelitian ini mengajukan pendekatan *pairwise* sebagai pendekatan yang dapat digunakan untuk melakukan perankingan informasi dalam kasus *Information Retrieval*, dimana dataset yang akan dimanaatkan adalah *dataset* LETOR 4.0 MQ2008. Dalam pendekatan *pairwise*, pasangan dokumen akan dianggap sebagai *input* untuk sistem pembelajaran yang tujuannya bukan untuk menentukan skor relevansi setiap dokumendengan kueri, tetapi dokumen mana yang lebih relevan daripada yang lain berdasarkan kueri. Selain itu, untuk meningkatkan performa model perankingan yang dibangun digunakan teknik penurunan gradien yang dapat meminimalkan nilai dari *loss function* model dengan memanfaatkan *decision tree* yaitu *gradient tree boosting algorithm*.

Berdasarkan hasil evaluasi NDCG yang diperoleh pada penelitian ini, penerapan algoritma *gradient tree boosting* dan pendekatan *pairwise* dalam kasus *Information Retrieval* dapat menghasilkan kualitas perankingan model yang baik.

UCAPAN TERIMA KASIH

Peneliti mengucapkan terima kasih kepada Tuhan Yang Maha Esa serta dosen pengampu mata kuliah Sistem Temu Balik Informasi yang telah mendukung dan membantu dalam memberikan kritik dan saran terkait penelitian ini.

REFERENSI

[1] J. Marcia Bates, *understanding Information Retrieval Systems*.: Auerbach Publications, 2011.

[2] A. Qurban Memon, *Distributed Networks*.: CRC Press, 2017.

[3] Alexey Grigorev, Jennifer L. Reese, and Richard M. Reese, *Java: Data Science Made Easy*.: Packt Publishing, 2017.

[4] Giuseppe Ciaburro and Prateek Joshi, *Python Machine Learning Cookbook - Second Edition*.: Packt Publishing, 2019.

[5] Butch Quinto, *Next-Generation Machine Learning with Spark: Covers XGBoost, LightGBM, Spark NLP, Distributed Deep Learning with Keras, and More*.: Apress, 2020.

[6] Hang Li, "A Short Introduction to Learning to Rank," *IEICE TRANS. INF. & SYST.*, vol. E94-D, 2011.

[7] Faiza Dammak, "Improving Pairwise Learning to Rank Algorithms for Document Retrieval," *IEEE*, 2017.

[8] Stefanie Molin, *Hands-On Data Analysis with Pandas*.: Packt Publishing, 2019.

[9] James Ma Weiming, *Mastering Python for Finance - Second Edition*.: Packt Publishing, 2019.

[10] P Li, "McRank: Learning to Rank Using Multiple Classification and Gradient Boosting," 2007.

[11] Tao Qin, Tie Yan Liu, Jun Xu, and Hang Li, "LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval," *Information Retrieval Journal*, August 2010.

[12] Yi Chang and Bo Long, *Relevance Ranking for Vertical Search Engines*.: Morgan Kaufmann, 2014.