Computers & Geosciences 142 (2020) 104520

Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

Improving search ranking of geospatial data based on deep learning using user behavior data

Yun Li [a,1], Yongyao Jiang [a,1], Chaowei Yang [a,*], Manzhu Yu [a], Lara Kamal [a], Edward M. Armstrong [b], Thomas Huang [b], David Moroni [b], Lewis J. McGibbney [b]

[a] NSF Spatiotemporal Innovation Center and Department of Geography and GeoInformation Science, George Mason University, Fairfax, VA, 22030, USA
[b] NASA Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109, USA

ARTICLE INFO

Keywords:
Deep learning
User behavior
Search engine
Knowledge discovery
Artificial intelligence

ABSTRACT

Finding geospatial data has been a big challenge regarding the data size and heterogeneity across various domains. Previous work has explored using machine learning to improve geospatial data search ranking, but it usually relies on training data labelled by subject matter experts, which makes it laborious and costly to apply to scenarios in which data relevancy to a query can change over time. When a user interacts with a search engine, plenteous information is recorded in the log file, which is essentially free, sustainable and up-to-the-minute. In this research, we propose a deep learning-based search ranking framework that can expeditiously update the ranking model through capturing real-time user clickstream data. The contributions of the proposed framework consist of 1) a log parser that can ingest and parse Web logs that record users' behavior in a real-time manner; 2) a set of hypotheses of modelling the relative relevance of data; and 3) a deep learning based ranking model which can be updated dynamically with the increment of user behavior data. Quantitative comparison with a few other machine learning algorithms suggests substantial improvement.

1. Introduction

Discovering geospatial data has been a big challenge for the data size and heterogeneity across various domains (Michael et al., 2018; Yang et al., 2017b). Because of the large amount of data, it is inefficient for users to discover desired information or documents by looking up the catalogue. As a result, creating an efficient, effective, and accurate geospatial search engine that retrieves related information from the Web has become more important than ever (Jiang et al., 2016b; Yang et al., 2017a).

In the past few years, many data portals have been built to allow users to better search, discover, visualize, refine, and access geospatial data. For example, Earthdata Search is one of such efforts that NASA developed to ease the technical burden on data users that makes it simple to interact with NASA Earth observation data (Reese et al., 2018). However, to truly free scientists from data discovery to spend more effort on innovation endeavors, more research remains to be done. One major problem with geospatial data search engines is the single attribute-based sorting (e.g. spatial resolution, popularity). This can be helpful sometimes, but it also restricts user's attention to one single data dimension. In reality, end users can have multiple and dynamic search preferences and trade-off needs to be made in order to have a balance of different data characteristics (Jiang et al., 2017a).

The primary goal is to develop an online deep learning framework to optimize the geospatial data discovery using user behavior data. The contributions of this framework consist of 1) a log parser that can ingest and parse real-time Web logs; 2) a set of hypotheses of modelling the relative relevance of data; and 3) a deep learning based ranking model which can be updated dynamically with the increment of user behavior data. We hope this research can strengthen ties between Earth observations and user communities by addressing the ranking challenge, an encumbrance in data discovery of a geospatial data portal.

* Corresponding author.
E-mail addresses: yli38@gmu.edu (Y. Li), yjiang8@gmu.edu (Y. Jiang), cyang3@gmu.edu (C. Yang), myu7@gmu.edu (M. Yu), lkamal3@gmu.edu (L. Kamal), Edward.M.Armstrong@jpl.nasa.gov (E.M. Armstrong), Thomas.Huang@jpl.nasa.gov (T. Huang), David.F.Moroni@jpl.nasa.gov (D. Moroni), Lewis.J.Mcgibbney@jpl.nasa.gov (L.J. McGibbney).

[1] Yongyao Jiang and Yun Li developed the methodology and solution; Chaowei Yang came up with the original idea and advised on methodologies; Manzhu Yu, Lara Kamel participated in the development of solutions, Edward M. Armstrong, Thomas Huang, and David Moroni did the validation and system test; Lewis J. McGibbney clearned the source code.

https://doi.org/10.1016/j.cageo.2020.104520
Received 14 October 2019; Received in revised form 6 May 2020; Accepted 13 May 2020
Available online 25 May 2020
0098-3004/

Oceanography data archived in NASA Physical Oceanography Distributed Active Archive Center (PO.DAAC) is selected as a proof of concept.

2. Literature review

Relevance ranking has been an active research area in fields including information retrieval, text mining, collaborative filtering (Yu et al., 2015; Severyn and Moschitti, 2015). Popular machine learning methods have been widely adopted to solve ranking problem named "learning-to-rank," a popular approach to solve information retrieval tasks. Learning to rank uses the training data, which consist of sets of query-document pairs along with relevance labels – ranging from very relevant to irrelevant - to learn a ranking model (Zong and Huang, 2014). In geospatial related domains, Martins and Calado (2010) applied a machine learning based ranking model algorithm to rank geospatial related newspaper documents. Hu et al. (2015) previously quantified the relevance of geospatial resources with a regression model. Shaw et al. (2013) introduced a spatial searching algorithm in which the machine learning method is leveraged to deduce users' geographic locations. As the most closely related work, Jiang et al. (2017a) proposed a machine learning based search ranking framework based on RankSVM, a Support Vector Machine (SVM) based ranking algorithm, for geospatial data discovery.

One shortcoming of traditional machine learning based ranking algorithms is that they must be trained in batch processing. In the process, a machine learning model is trained on labelled data and periodically updated to predict relevance scores of unseen samples, which can be utilized by the ranking algorithm (Moon et al., 2010; Jiang et al., 2018; Burges et al., 2005). However, there are at least three drawbacks of these traditional machine learning algorithms such as SVM. First, the time and computation resources cost on training a model can increase over time. Second, it assumes the data relevance to a certain query is almost static over time. Third, machine learning based ranking algorithms typically requires training data originated from relevance judgments provided by subject matter experts, which makes them laborious and costly to apply. Thus, automatic training data preparation is critical for the machine learning based ranking, which saves time and enables dynamic data relevance to queries.

As a new paradigm of machine learning, deep learning is capable of discovering implicit knowledge in high-dimensional data and has achieved remarkable success in many domains of science, business and government (Lecun et al., 2015), including geospatial domains (Li et al. 2015, 2017; Song et al., 2015). One of the typical algorithms is the deep artificial neural network, whose nature allows it to be trained in a sequential order and update the weights using incoming data at each step. In contrast, traditional batch training techniques train a model by fitting the entire training data set at once. Therefore, we propose to apply deep learning to overcoming the first limitation of batch training of traditional machine-learned ranking algorithms.

While the learning algorithm is important, how to extract quality training data is also crucial. Most users do not click links randomly, but do selections based on their domain knowledge. Although user behavior data is not perfect and can be noisy, the aggregated click patterns are likely to reveal some information about data relevancy. Although a series of works have been proposed to extract training data from user clicking behavior automatically (Joachims, 2002; Chapelle and Zhang, 2009; Sontag et al., 2012), most of them demand an initial ranking list displayed to users. The initial ranking results shown to users is challenging to deduce and restore just from Web logs and in practice they are generally recorded by a pre-configured third-party front/back end software. I propose two hypotheses of generating training data from user data to address the second and third limitation of traditional machine-learned ranking algorithms. When a user interacts with a search engine, a great deal of information about data relevancy is recorded in logs. Compared to explicit feedbacks, information extracted from user behavior logs is essentially free, sustainable and up-to-the-minute (Gashaw and Liu, 2017).

Web log processing, which serves the purpose of reconstructing each single user visit from raw Web logs, is an important prior step of improving geospatial data discovery. Most existing research deals with this problem by using an empirical and fixed timeout threshold to split sessions (Neelima and Rodda, 2016). However, the threshold is highly dependent on site structures and user groups. Others have proposed a time-based hierarchical clustering method (Jiang et al., 2016a). Although there is a faster large-scale in-memory version of this cluster method has been proposed (Li et al., 2016; Jin et al., 2017), it mostly can only be applied in an off-line manner. We therefore propose a real-time log processing workflow that can ingest and process web logs to extract training data in a real-time fashion.

Compared with our previous work (Jiang et al., 2017a), the main contribution of this paper contains (1) a deep learning based ranking algorithm that can be updated in real time in contrast to RankSVM that can only be training in batch mode and won't be able to handle incremental training, which is known to be best by deep learning for continuously improving accuracy; (2) an approach to derive training data from user behavior; (3) a real-time log processing pipeline to ingest and extract user behavior from Web logs.

3. Methodology

3.1. Framework architecture

Our deep learning-based search ranking framework includes four major components: semantic similarity calculator, log processer, feature extractor, and deep learning based ranking algorithm (Fig. 1). Given a user query, it first gets transformed into a semantic query through the semantic similarity calculator. For instance, "*sea surface salinity*" would be translated into "*sea surface salinity OR sss.*" More detail can be found at Jiang et al. (2017b). The top K related documents to the semantic query would then get returned. Once these top K results are obtained, ranking features would be extracted for each of the search results on the fly, which will be discussed further in the next section. After that, the top K results would be re-ranked by putting into a pre-trained deep learning based ranking model. While users interact with the search engine user interface, new logs would be generated, and the log processer would then extract new training data to update the ranking model in real-time or on a regular basis.

3.2. Ranking features

Ranking functions in most geospatial search engines only consider

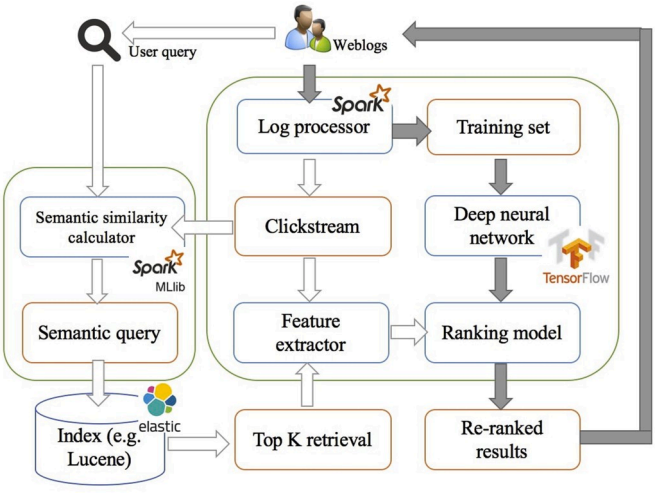

Fig. 1. Deep learning ranking framework.

term frequency. In other words, the more frequent a query term appears in the dataset metadata, the more relevant this dataset is. However, after discussing with domain experts, a lot of more other factors are taken into account when selecting the most appropriate. For instance, release date is a very important factor as more recent data tends to be more accurate in measurements (e.g. sea surface temperature). We therefore identified seventeen features that can be categorized into query-dependent and query-independent features. Query-dependent features refer to features that are determined by both the data and the query, while query-independent are not associated with any specific query. More specifically, query-dependent features include text-based similarity, spatial similarity and query-dependent popularity. Query-independent features include release date, version number, data processing level, spatial and temporal resolution, and query-independent popularity (Jiang et al., 2017a).

3.3. Online log processing

An online log processing workflow is designed to split a collection of web logs into sessions, each of which represents behaviors during one visit of a single user. When a new web log comes in, a new session is created if this user does not exist in all the ongoing sessions cached in memory (Fig. 2). A unique user is determined by the combination of IP address and user agent/browser. Otherwise, if its time interval with the last log of the ongoing session is less than a threshold, it joins the ongoing session. If the time interval is greater than the threshold, the ongoing session ends, and a new session starts. In addition, if an ongoing session does not receive any new logs longer than the time threshold, it also ends. Whenever a session ends, it goes through the crawler detection process that removes web logs generated by web crawler to the largest extent based on user agent, robot.txt, request sending rate (Jiang et al., 2016a) in three steps:

- For well-known search engine crawlers, we can use their identities they use in user-agent field by maintaining a list of known crawlers.
- For other "well-behaved" crawlers which abide by standard robot exclusion protocols, begin their site crawl by first attempting to access the exclusion file "robots.txt" in the server root directory. Therefore, we can identify those by checking whether a request to the robots.txt file was made.
- For many crawlers not in the previous categories, we examine other two important features: maximum sustained request rate and the number of request types by checking if they are beyond an upper bound on the maximum number of clicks that a human can make within a specific time frame.

The filtered session results are then connected with referrer/previous page information.

The online log processer is implemented with Spark Streaming APIs which support scalable and efficient stream processing of live data such as Web logs. Data can be ingested from a single file, HDFS, or advanced messaging software like Kafka. Internally, Spark Streaming APIs ingests streaming data and split them into small batches. Then, these small batches are get transformed to generate the live processing results in batches (Fig. 3). Spark streaming handles one batch at a time. Additionally, the individual data items within each batch are processed in their order within the batch. By default, if spark doesn't have enough time to get to all the data items in a batch when the next one comes, those data items will be dropped. Therefore, it is important to balance the batch interval and batch processing time to prevent data loss.

3.4. Creating training data from user session

Fig. 4 describes a typical data search scenario. After users put in some keywords, they click on a few data in the search results. After that, they add a search filter (e.g. processing level) and click on some data again. Eventually, they end up downloading some data. One may guess dependencies exist between query, filter, viewed data, and downloaded data. While this kind of dependency is fascinating for analysis, it can also involve lots of noise. For example, the fact that data 2 is clicked after data 1 does not suggest that data 2 is more relevant than data 1. Studies have also shown that users are less likely to click on an item that is low in the ranked list, regardless of how relevant the item is to the search query. In an extreme situation, the probability turns to nearly zero for a link at rank 10.000 to be clicked even though the link is very relevant to the query (Langville et al., 2008). Therefore, a view on a data cannot be regarded as an absolute relevance judgment, and it is more plausible to make inference about relative relevance judgment. In this context, two hypotheses of modelling the relative judgments are proposed. Since users' behaviors can be quite complicated, we simplified the modelling process and summarized them into two hypotheses. In the hypotheses, symbols ">" and "<" represent the level of relevancy between two data to the same query. Data in the left side of ">" is more relevant to a query than that of the data in the right side, and vice versa for "<".

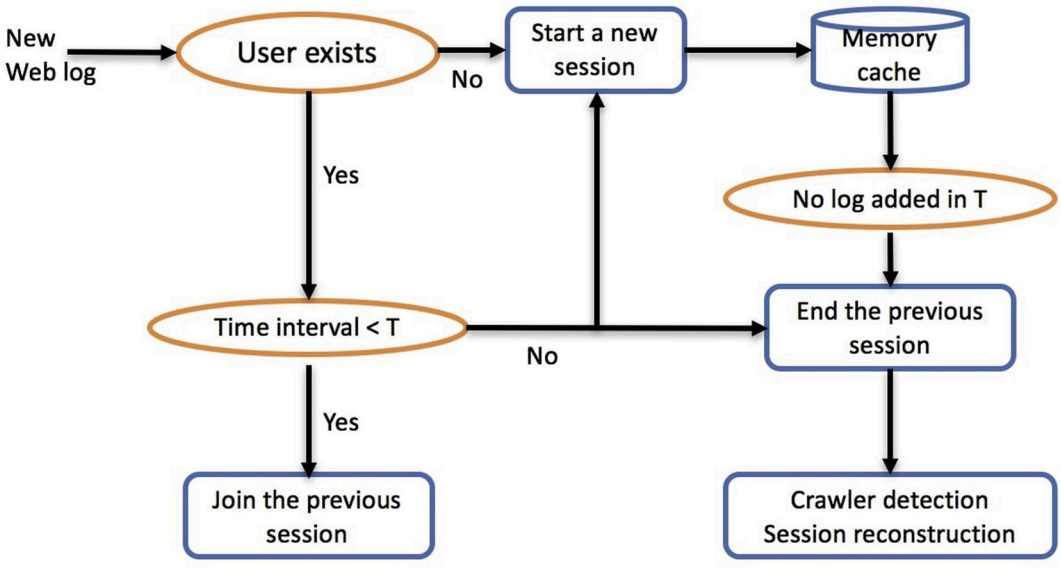

Fig. 2. Workflow of the online processing.

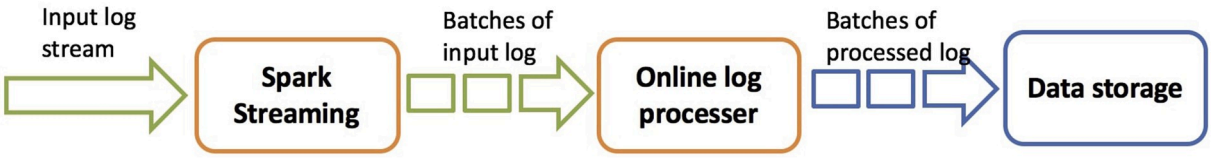

Fig. 3. Implementation of the online log processer.

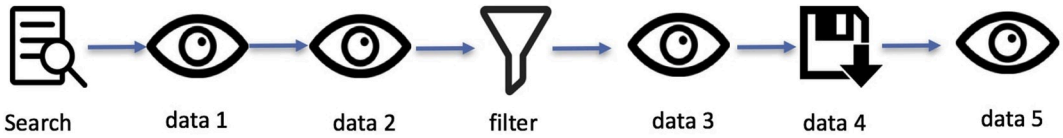

Fig. 4. Typical user search scenario.

H1. Downloaded data are more relevant than the data viewed before. Take Fig. 4 as an example, data 1, data 2, and data 3 < data4.

H2. In addition to H_1, the filtered results are more relevant that the data viewed before. The rationale behind this is that the reason users click on a specific filter is because they prefer those filtered results. In this case, data 1, data 2, and data 3 < data4; data 1 and data 2 < data 3; data 1 and data 2 < data 5. No inference of relevancy is made between data 3 and data 5, data 1 and data 2.

For example, Fig. 5 illustrates a session extracted from PO.DAAC logs, where the user:

- viewed sles_l2_jason2_v1 (d_1) after searching sea surface topography.
- added a filter processing level 4 to the query
- viewed recon_sea_level_ost_l4_v1 (d_2)
- downloaded alt_tide_gauge_l4_ost_sla_us_west_coast (d_3).

According to H_1, $d_3 > d_1$, $d_3 > d_2$. According to H_2, $d_2 > d_1$ is also extracted. Comparing H_1 and H_2, experiments in the result section found that more training data can be extracted by applying H_2, but more noise will also be introduced to the training data.

3.5. Deep learning based ranking algorithm

This ranking algorithm consists of three major steps:

Step 1: Calculate the difference between each data pair. The goal of this step is to convert the ranking problem into a binary classification. As shown in Fig. 6, the left table shows the original feature vectors, and the right one lists the feature difference vector. For each pair $d_i - d_j$, if data i is less relevant than data j, a negative label is assigned, reversely, a positive label is applied. Through this transformation, we get to work with the difference vectors of all possible data pairs rather than directly deal with the initial feature vectors (e.g., d_1 *and* d_2).

Step 2: Apply a deep neural network to the transformed dataset and calibrate all weights by repeating two key steps, forward propagation and back propagation (Fig. 7). Though we used four hidden layers, the figure below only uses one hidden layer to illustrate the idea.

There are usually three different training strategies when fitting a neural network (Fig. 8). Stochastic Gradient Descent (SGD) fits neural network using one sample at a time. Mini-batch gradient descent fits it with one small group of samples at a time, while the regular gradient descent trains the network using the entire dataset each time. As we can see, although SGD and mini-batch go through many oscillations to converge, they can both eventually converge and are a lot faster to compute as they require less data. The kind of nature of neural network makes it possible to implement a real-time ranking model which can update itself as users interact with search engines. A bonus that comes with this implementation is that the ranking function can then adapt to users' changing search preferences.

Step 3: Implement a sorting algorithm (e.g. quicksort) to transform the neural network prediction back to an ordered list and present to users.

4. Experiment

In the experiment, we used all publicly available metadata from PO.DAAC. These metadata describe data of over 30 variables ranging from sea surface temperature, circulation, salinity to ocean wind and gravity.

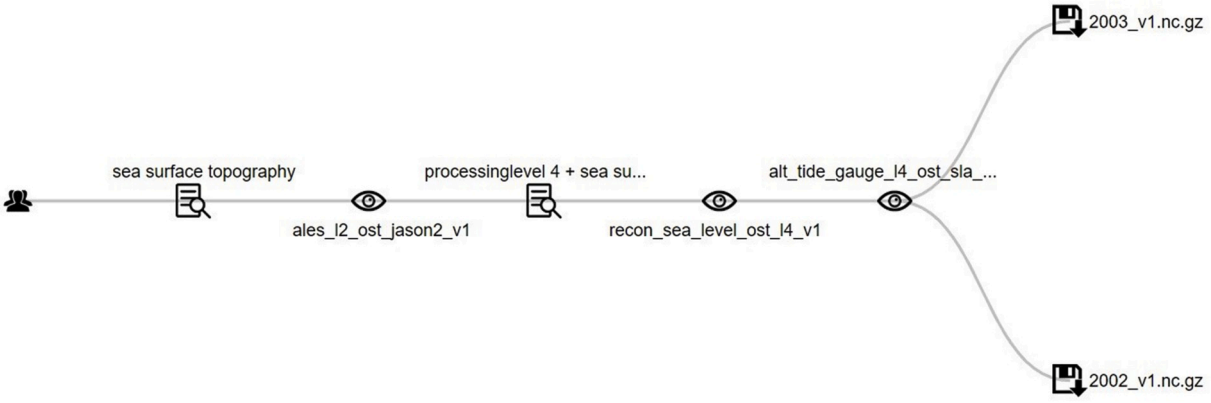

Fig. 5. Usage scenario for applying H1 and H2.

Data	Feature_1	Feature_2	Action
d_1	0.9	0.8	view
d_2	0.4	0.7	view
d_3	0.8	0.6	download

Data	Feature_1	Feature_2	Label
d_3-d_1	-0.1	-0.2	+1
d_2-d_3	-0.4	0.1	-1

Fig. 6. Concrete example of pairwise transformation.

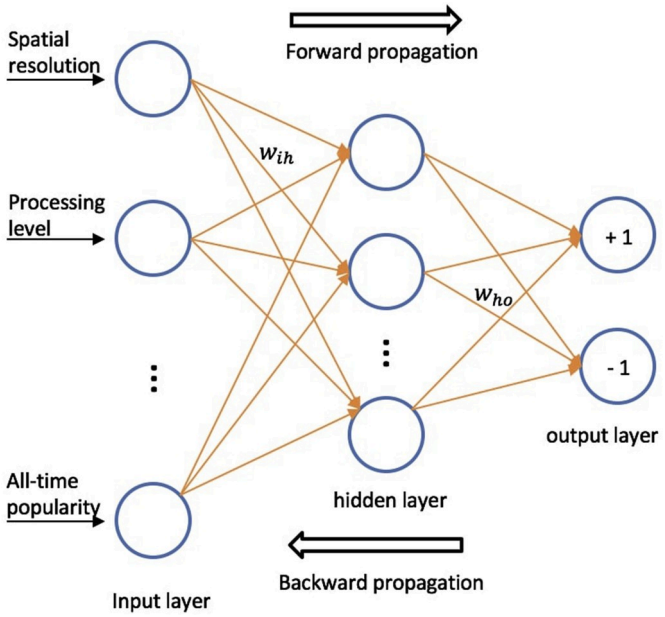

Fig. 7. Artificial neural network.

As for user behavior, five years of Web logs (600 million records, 150G) from PO.DAAC data search engine were used. These web logs can be seen as structured data that contains fields such request time, IP address, page requested, HTTP code, user agent, referrer, and bytes served (Jiang et al., 2016a).

The proposed deep learning ranking algorithm was compared against a few other machine learning algorithms by replacing the algorithm used in step 2 (e.g. KNN, logistic regression). Data derived from Web logs based on H_1 and H_2 were randomly split into training data and validation data. Fifty queries were randomly selected from the web logs such as *"sea surface temperature"*, *"ocean waves"*, *"gravity"*, and *"saline density AND Atlantic Ocean"*, etc. Each query has about 80 search results on average. Validation dataset was created by a few ocean scientists and graduate students who labelled the retrieval results of each query manually using a seven-point scale ranging from 0 to 7 to indicate how relevant a data is.

5. Results and evaluation

5.1. Online learning curve

Fig. 9 shows how loss and accuracy changes during the mini-batch process. Blue line represents the training data derived from H_1 and orange line is the testing data (i.e. human labelled data). As we iterate through the training process, the loss of testing data keeps going down and the accuracy keeps going up, while there is oscillation with training data. This result demonstrates that the accuracy can indeed converge during the mini-batch training process. In this case, the accuracy eventually converges at 82.06%.

5.2. Comparison of H_1 and H_2

Table 1 shows the accuracy of H_1 and H_2 with five different machine learning algorithms. These accuracies are the best we could achieve using grid search and hyper-parameter tuning. As can be seen, the accuracy of H_1 is generally better than H_2. A possible explanation is that H_2 introduces more noise although it produces more training data. In the meanwhile, as the most flexible non-linear learning algorithm, DNN outperforms all the other algorithms with H_1. Combined with the support for SGD and mini-batch training, it justifies the selection of deep neural network for this work.

5.3. Precision at K

Precision at K (P(K)) as a well-recognized evaluation metric for ranking was also used to measure how the proposed algorithm performs over the ranked testing data. P(K) essentially measures the ratio of relevancy of the top K data in the rank list. A relevant data is required to have a relevance score above "3" is our specific setting. As presented in Fig. 10, DNN demonstrates significant improvement over the other ranking methods. Although KNN tends to have similar performance after position 5, DNN perform better for the first five positions which are the most important in the ranking scenario.

5.4. A concrete example

Fig. 11 compares the search results of "ocean OR wind" in PO.DAAC search engine and MUDROD, the Web interface we have developed. PO.DAAC retrieves 382 matches while MUDROD has 471 matches. The first two search results on PO.DAAC search engine are about ocean temperature and sea surface topography (the orange words suggest dataset's

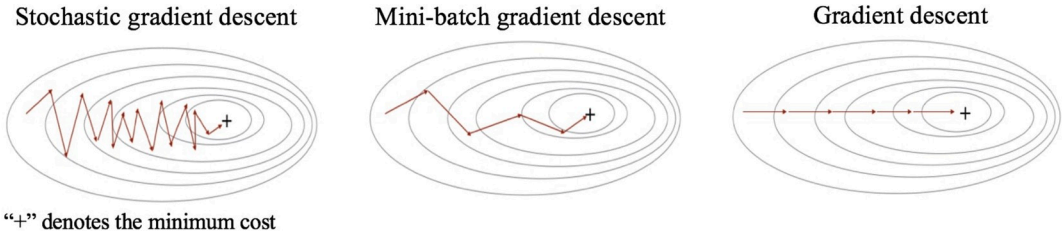

Fig. 8. Neural network training strategies.

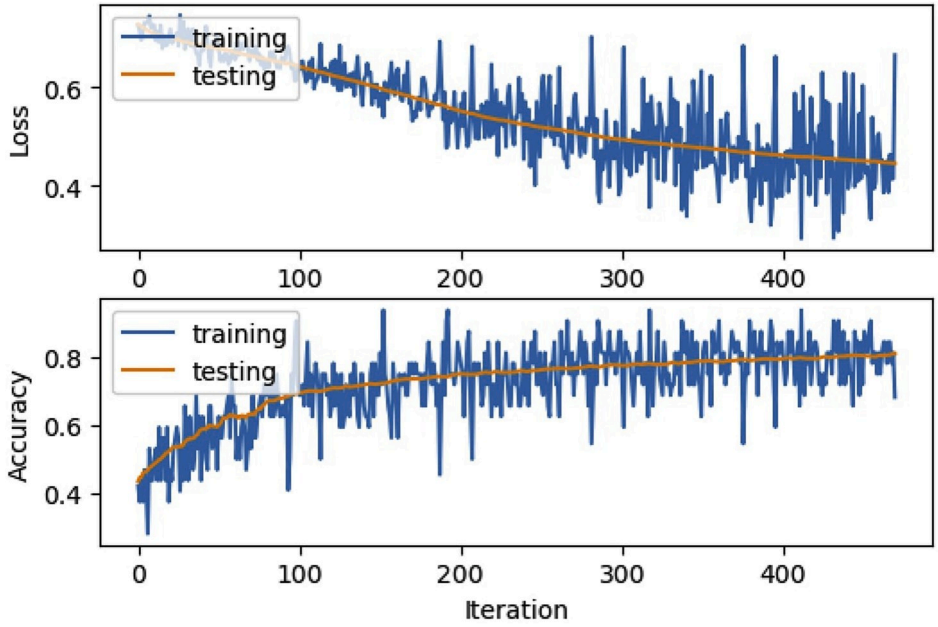

Fig. 9. How accuracy changes during the mini-batch training.

Table 1
Comparison of two hypotheses' accuracy.

	H1	H2
DNN	82.06%	67.61%
Logistic regression	79.67%	64.27%
SVM	80.76%	69.34%
KNN	74.55%	63.26%
Random forest	80.37%	71.62%

topics), while most results on MUDROD are about ocean wind except the second one. The example suggests improvements of our system with regard to both recall and precision.

6. Conclusion and discussion

This article reports our work to tackle a crucial challenge in geospatial data discovery, the ranking problem, using deep learning and user behavior data, in other words, the wisdom of crowd. When a user interacts with a search engine, plenteous information is recorded in the log file, which is essentially free, sustainable, and up-to-the-minute.

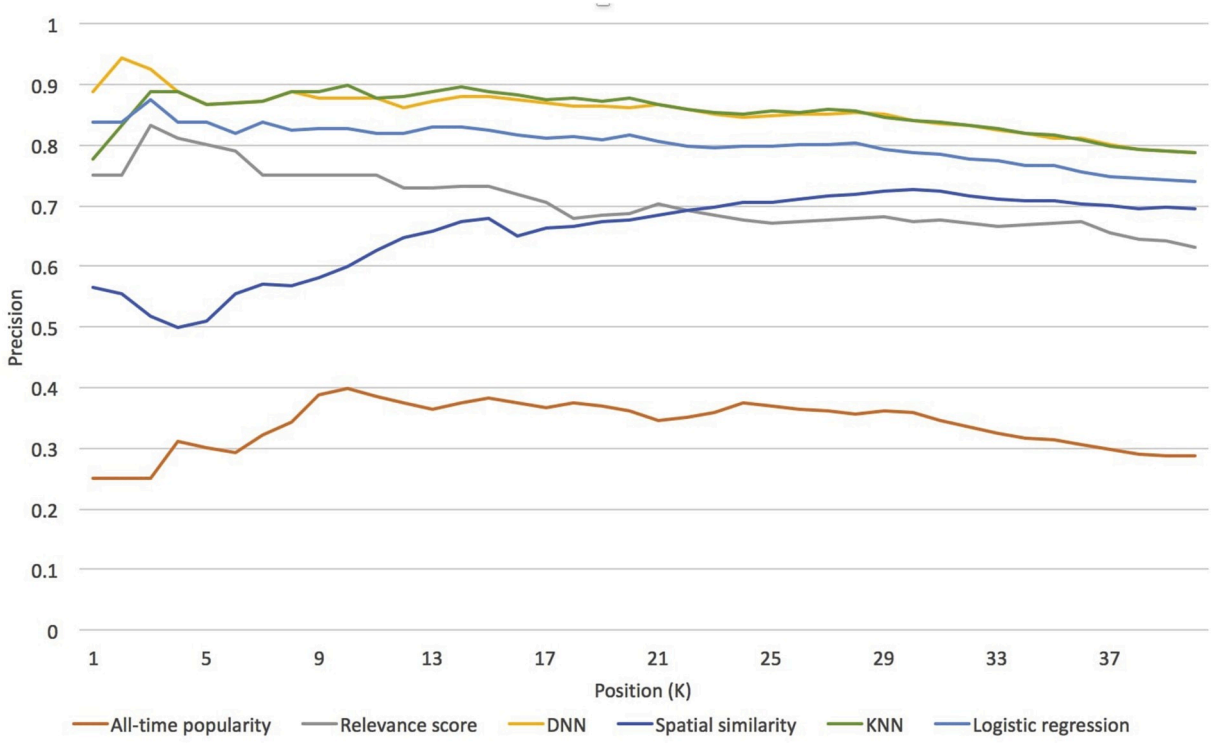

Fig. 10. Evaluation with precision at K.

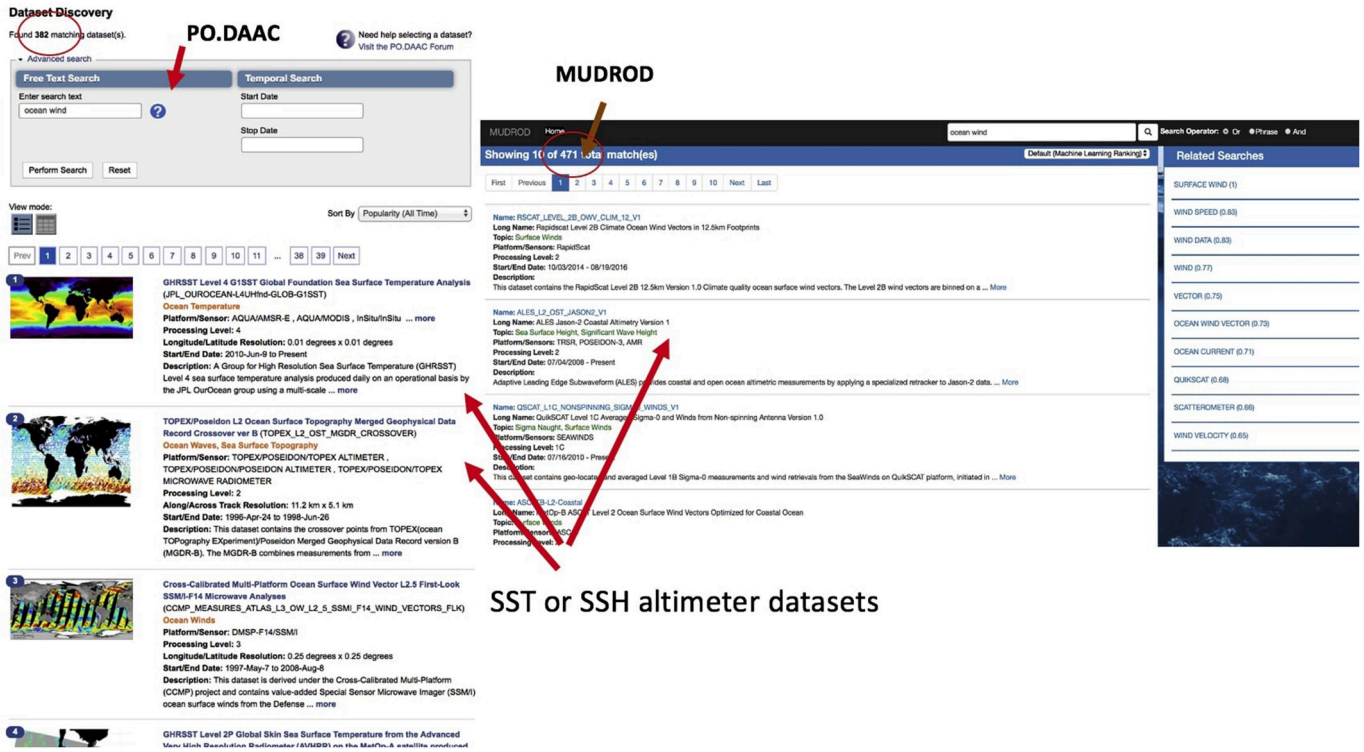

Fig. 11. Comparison of "ocean OR wind" search results on two search engines.

Experiment has shown that the log processing workflow can successfully, automatically and efficiently produce training data from real-time Web streaming logs. Specifically, the evaluation shows H_1 tend to produce more reliable and stable training data than H_2. The proposed deep learning ranking algorithm shows significant improvements over the existing machine learning ranking techniques.

There are a few directions where this research can be improved. First, we are going to add more ranking features and conduct more sophisticated feature engineering in order to further improve the model accuracy. Second, we plan to investigate the search engine bias issue - "winner-take all" effect originated from top placement in a search list. Some literature pointed out it can be mitigated through personalized search (Goldman, 2008). In addition, we are exploring how to do ranking with the proposed algorithm in scale, in other words, how do conduct efficient ranking with millions of search results (Jin et al., 2017). The performance of the method for large training datasets should be improved to leverage the advantages of big training datasets for further improving the results (Yang et al., 2019). To improve the usability of the tools developed, we are also planning to produce a docker/container image that can be readily spun off by anyone who are familiar with the latest cloud computing technologies (Yang et al., 2017c).

Computer code availability

All source code developed is open source and available at GitHub site https://github.com/stccenter/MUDROD.

Acknowledgements

Research reported is supported by NSF (1835507, 1841520) and NASA (NNX15AM85G).

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.cageo.2020.104520.

References

Burges, C., et al., 2005. Learning to rank using gradient descent. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 89–96.
Chapelle, O., Zhang, Y., 2009. A dynamic bayesian network click model for web search ranking. In: Proceedings of the 18th International Conference on World Wide Web, pp. 1–10.
Gashaw, Y., Liu, F., 2017. Performance evaluation of frequent pattern mining algorithms using web log data for web usage mining. In: Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2017 10th International Congress on, pp. 1–5.
Goldman, E., 2008. Search Engine Bias and the Demise of Search Engine Utopianism. Web Search. Springer, pp. 121–133.
Hu, Y., et al., 2015. Metadata topic harmonization and semantic search for linked-data-driven geoportals: a case study using ArcGIS online. Trans. GIS 19 (3), 398–416.
Jiang, Y., et al., 2016a. Reconstructing sessions from data discovery and access logs to build a semantic knowledge base for improving data discovery. ISPRS Int. J. Geo-Inf. 5 (5), 54.
Jiang, Y., et al., 2017a. Towards intelligent geospatial data discovery: a machine learning framework for search ranking. Int. J. Dig. Earth 1–16.
Jiang, Y., et al., 2018. A smart web-based geospatial data discovery system with oceanographic data as an example. ISPRS Int. J. Geo-Inf. 7 (2), 62.
Jiang, Y., et al., 2017b. A comprehensive methodology for discovering semantic relationships among geospatial vocabularies using oceanographic data discovery as an example. Int. J. Geogr. Inf. Sci. 31 (10), 19.
Jiang, Y., et al., 2016b. Polar CI portal: a cloud-based polar resource discovery engine. In: Cloud Computing in Ocean and Atmospheric Sciences, pp. 163–185.
Jin, B., et al., 2017. A high performance, spatiotemporal statistical analysis system based on a Spatiotemporal Cloud Platform. ISPRS Int. J. Geo-Inf. 6 (6), 165.
Joachims, T., 2002. Optimizing search engines using clickthrough data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 133–142.
Langville, A.N., Meyer, C.D., Fernández, P., 2008. Google's pagerank and beyond: the science of search engine rankings. Math. Intel. 30 (1), 68–69.
Lecun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.
Li, S., et al., 2015. Using CA-Markov model to model the spatiotemporal change of land use/cover in Fuxian Lake for decision support. ISPRS Ann. Photogr. Remote Sens. Spatial Info. Sci. 2 (4), 163.
Li, Y., et al., 2016. Leveraging Cloud Computing to Speedup User Access Log Mining. In: OCEANS 2016 MTS/IEEE Monterey, pp. 1–6.
Li, Y., et al., 2017. Leveraging LSTM for rapid intensifications prediction of tropical cyclones. In: ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences, 4.

Martins, B., Calado, P., 2010. Learning to rank for geographic information retrieval. In: Proceedings of the 6th Workshop on Geographic Information Retrieval, p. 21.
Michael, R., Jonathan, B., David, S., 2018. Enabling discovery of ocean science data in the modern era using geospatial and information science. In: BOOK OF ABSTRACTS.
Moon, T., et al., 2010. Online learning for recency search ranking using real-time user feedback. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp. 1501–1504.
Neelima, G., Rodda, S., 2016. Predicting user behavior through sessions using the web log mining. In: Advances in Human Machine Interaction (HMI), 2016 International Conference on, pp. 1–5.
Reese, M., Lynnes, C., Newman, D., 2018. End-to-End Solution for Data Customization with NASA's Earthdata Search.
Severyn, A., Moschitti, A., 2015. Learning to rank short text pairs with convolutional deep neural networks. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 373–382.
Shaw, B., et al., 2013. Learning to rank for spatiotemporal search. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, pp. 717–726.
Song, J., Wu, J., Jiang, Y., 2015. Extraction and reconstruction of curved surface buildings by contour clustering using airborne LiDAR data. Optik-Int. J. Light Electron Opt. 126 (5), 513–521.
Sontag, D., et al., 2012. Probabilistic models for personalizing web search. In: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, pp. 433–442.
Yang, C., et al., 2017a. Utilizing cloud computing to address big geospatial data challenges. Comput. Environ. Urban Syst. 61, 120–128.
Yang, C.P., et al., 2017. An architecture for mitigating near earth object's impact to the earth. In: Aerospace Conference, 2017. IEEE, pp. 1–13.
Yang, C., et al., 2017c. Big Data and cloud computing: innovation opportunities and challenges. Int. J. Dig. Earth 10 (1), 13–53.
Yang, C., et al., 2019. Big Earth data analytics: a survey. Big Earth Data 3 (2), 83–107.
Yu, J., et al., 2015. Learning to rank using user clicks and visual features for image retrieval. IEEE Trans. Cybernet. 45 (4), 767–779.
Zong, W., Huang, G.-B., 2014. Learning to rank with extreme learning machine. Neural Process. Lett. 39 (2), 155–166.