The Random Neural Network Applied to an Intelligent Search Assistant

Will Serrano(✉)

Intelligent Systems and Networks Group Electrical and Electronic Engineering, Imperial College London, London, UK
g.serranoll@imperial.ac.uk

Abstract. Users can not guarantee the results they obtain from Web search engines are exhaustive, or that they actually respond to their needs. Search results are influenced by the users' own ambiguity in formulating their requests or queries as well as by the commercial interest of Web search engines and Internet users that want to reach a wider audience. This paper presents an Intelligent Search Assistant (ISA) based on a Random Neural Network that acts as the interface between users and search engines to present data to users in a manner that reflects their actual needs or their observed or stated preferences. Our ISA tracks the user's preferences and makes a selection on the output of one or more search engines using the preferences that it has learned. We also introduce a "relevance metric" to compare the performance of our Intelligent Search Assistant against a few search engines, showing that it provides better performance.

Keywords: Intelligent search assistant · World wide web · Random neural network · Web search · Search engines

1 Introduction

Web Search Engines have been used as the direct connection between users and the information or products sought in the Internet. Search results are influenced by a commercial interest as well as by the users' own ambiguity in formulating their requests or queries. Ranking algorithms are essential in Web search as they decide the relevance; they make information visible or hidden to customers or users. Under this model, Web search engines or recommender systems can be tempted to artificially rank results from some specific businesses for a fee whereas also authors or business can be tempted to manipulate ranking algorithms by "optimizing" the presentation of their work or products. The main consequence is that irrelevant results may be shown on top positions and relevant ones "hidden" at the very bottom of the search list.

In order to address the presented search issues; this paper proposes an Intelligent Search Assistant (ISA) that acts as an interface between an individual user's query and the different search engines. Our ISA acquires a query from the user and retrieves results from one or various search engines assigning one neuron per each Web result dimension. The result relevance is calculated by applying our innovative cost function based on the division of a query into a multidimensional vector weighting its dimension

T. Czachórski et al. (Eds.): ISCIS 2016, CCIS 659, pp. 39–51, 2016.
DOI: 10.1007/978-3-319-47217-1_5

terms with different relevance parameters. Our ISA adapts and learns the perceived user's interest and reorders the retrieved snippets based in our dimension relevant centre point. Our ISA learns result relevance on an iterative process where the user evaluates directly the listed results. We evaluate and compare its performance against other search engines with a new proposed quality definition, which combines both relevance and rank. We have also included two learning algorithms; Gradient Descent learns the centre of relevant dimensions and Reinforcement Learning updates the network weights based on rewarding relevant dimensions and punishing irrelevant ones. We have validated our ISA against other Web search engines using travel services and open user queries. We have also analysed the Gradient Descent and Reinforcement Learning algorithms based on result relevance and learning speed.

We describe the application of neural networks in Web search in Sect. 2. We define our Intelligent Search Assistant mathematical model in Sect. 3 and we have validated it against other Web search engines in Sect. 4. Finally, we present our conclusions in Sect. 5.

2 Related Work

Neural networks have been already applied in the World Wide Web as a mechanism of adaptation to users' interest in order to provide relevant answers. Wang et al. [1] use a back propagation neural network with its input nodes corresponding to an specific quantified user profile and one output node which it is the a probability the user would consider the Web page relevant. Boyan et al. [2] use reinforcement learning to rank Web pages using their HTML properties and hyperlink connections between them. Shu et al. [3] retrieve results from different Web search engines and train the network following the assumption that a result in a top position would be relevant. Burgues et al. [4] define RankNet which uses neural networks to evaluate Web sites by training the neural network based on query-document pairs. Bermejo et al. [5] use a similar approach to our proposal, the allocation of one neuron per Web search result, however the main difference is that the network is trained to cluster results by meaning. Scarselli et al. [6] use a neural network by assigning a neuron to each Web page; they create a graph where the neural links are the equivalent of the hyperlinks.

3 The Intelligent Search Assistant Model

The search assistant we design is based on the Random Neural Network (RNN) [7–9, 19]. This is a spiking recurrent stochastic model for neural networks. Its main analytical properties are the "product form" and the existence of the unique network steady state solution. The RNN represents more closely how signals are transmitted in many biological neural networks where they actual travel as spikes or impulses, rather than as analogue signal levels. It has been used in different applications including network routing with cognitive packet networks [10], search for exit routes for evacuees in emergency situations [11, 12], pattern based search for specific objects [13], video compression [14], and image texture learning and generation [15].

3.1 Search Model

In the case of our own application of the RNN, the search for information or for some meaning requires us to specify: an M-dimensional universe of X entities or ideas to be searched, a high level query that specifies the N-properties or concepts requested by a user and a method that searches and selects Y entities from the universe showing the first Z results to user according to an algorithm or rule. Each entity or concept in the universe is distinct from the others in some recognizable way; for instance two entities may be different just in the date or time-stamp that characterizes the time when they were last stored or in the ownership or origin of the entities. On the other hand, we consider concepts to be distinct if they contain any different meaning, even though if they are identical with respect to a user's query.

We consider that the universe which we are searching within as a relation U that consists of a set of X M-tuples, $U = \{v_1, v_2 \ldots v_X\}$, where $v_i = (l_{i1}, l_{i2} \ldots l_{iM})$ and li are the M different attributes for i = 1, 2..X. The relation U is a very large relation consisting on $M \gg N$ attributes. The important concept in the development of this paper is a query can be defined as $R_t(n(t)) = (R_t(1), R_t(2), \ldots, R_t(n(t)))$ where n(t) is a variable N-dimension attribute vector with $1 < N < M$ and t is the search iteration being $t > 0$; n (t) is variable so that attributes can be added or removed based on their relevance as the search progresses, i.e. as t increases. Each $R_t(n(t))$ takes its values from the attributes within the domain D(n(t)), where D is the corresponding domain that forms the universe U. Thus D(n(t)) is a set of properties or meanings based in words or integers, but also words in another language, or a set of icons, images or sounds.

The answer A to the query $R_t(n(t))$ is a set of Y M-tuples $A = \{v_1, v_2 \ldots v_Y\}$ where $v_o = (l_{o1}, l_{o2} \ldots l_{oM})$ and lo are the M different attributes for o = 1, 2..Y. Our Intelligent Search Assistant only shows to the user the first set of Z tuples that have the highest neuron potentials among the set of Y tuples. The neuron potential that represents the relevance of each M-tuple v_o is calculated at each t iteration. The user or the high level query itself is limited mainly by two main factors: the user's lack of information about all the attributes that form the universe U of entities and ideas, or the user's lack of precise knowledge about what he is looking for.

3.2 Result Cost Function

We consider the universe U is formed of the entire results that can be searched. We assign each result provided by a search engine to an M-tuple v_o of the answer set A. We calculate the result relevance based on a cost function described within this section. The query $R_t(n(t))$ is a variable N-dimension vector that specifies the attributes the user consider relevant. The number of dimensions of the attribute vector n(t) varies as the iteration t increases. Our Intelligent Search Assistant associates an M-tuple v_o to each result provided by the Search Engine creating an answer set A of Y M-tuples. Search Engines select their results from the universe U. We apply our cost function to each result or M-tuple v_o from the answer set A of Y M-tuples. We consider each v_o as a M-dimensional vector. The cost function is firstly calculated based on the relevant N attributes the user introduced on the High Level Query, $R_1(n(1))$ within the domain

D(n(1)) however, as the search progresses, $R_t(n(t))$, attributes may be added or removed based on the perceived relevance within the domain D'(n(t)). We calculate the overall Result Score, RS, by measuring the relationship between the values of its different attributes:

$$RS = RV * HW \tag{1}$$

where RV is the Result Value which measures the result relevance and HW the Homogeneity Weight. The Homogeneity Weight (HW) rewards results that have relevance or scores dispersed along their attributes. This parameter is also based on the idea that the first dimensions or attributes of the user query $R_t(n(t))$ are more important than the last ones:

$$HW = \frac{\sum_{n=1}^{N} HF[n]}{N} \tag{2}$$

where HF[n], homogeneity factor, is a N-dimension vector associated to the result and n is the attribute index from the query $R_t(n(t))$:

$$HF[n] = \begin{cases} \frac{N-n}{N} & \text{if } SD[n] > 0 \\ 0 & \text{if } SD[n] = 0 \end{cases} \tag{3}$$

We define Score Dimension SD[n] as a N-dimension vector that represents the attribute values of each result or M-tuple v_o in relation with the query $R_t(n(t))$. The Result Value (RV) is the sum of each dimension individual score:

$$RV = \sum_{n=1}^{N} SD[n] \tag{4}$$

where n is the attribute index from the query $R_t(n(t))$. Each dimension of the Score Dimension vector SD[n] is calculated independently for each n-attribute value that forms the query $R_t(n(t))$:

$$SD[n] = S * PPW * RPW * DPW \tag{5}$$

We consider only three different types of domains of interest: words, numbers (as for dates and times) and prices. S is the score calculated depending if the domain of the attribute is a word (WS), number (NS) or price (PS). If the domain D(n) is a word, our ISA calculates the score Word Score (WS) following the formula:

$$S = \frac{WR}{NW} \tag{6}$$

where the value of WR is 1 if the word of the n-attribute of the query $R_t(n(t))$ is contained in the search result or 0 otherwise. NW is the number of words in the search

result. If the domain D(n) is a number, our ISA selects the best Number Score (NS) from the numbers they are contained within the search result that maximizes the cost function:

$$S = \frac{\left(1 - \left(\frac{|DV - RV|}{|DV| + |RV|}\right)\right)}{NN} \tag{7}$$

where DV is the value of the n-attribute of the query $R_t(n(t))$, RV is the value of a number in the result and NN is the total number of numbers in the result. If the domain D(n) is a price, our ISA chooses the best Price Score (PS) from the prices in the result that maximizes the cost function:

$$S = \frac{\left(\frac{DV}{RV}\right)}{NP} \tag{8}$$

where DV is value of the n-attribute of the query $R_t(n(t))$, RV is the value of a price in the result and NP is the total number of prices in the result. We penalize if the search result provides unnecessary information by dividing the score by the total amount of elements in the Web result. The dimension Score Dimension vector, SD[n] is weighted according to different relevance factors:

$$SD[n] = S * PPW * RPW * DPW \tag{9}$$

The Position Parameter Weight (PPW) is based on the idea that an attribute value shown within the first positions of the search result is more relevant than if it is shown at the final:

$$PPW = \frac{NC - DVP}{NC} \tag{10}$$

where NC is the number of characters in the result and DVP is the position within the result where the value of the dimension is shown. The Relevance Parameter Weight (RPW) incorporates the user's perception of relevance by rewarding the first attributes of the query $R_t(n(t))$ as highly desirable and penalising the last ones:

$$RPW = 1 - \frac{PD}{N} \tag{11}$$

where PD is the position of the n-attribute of the query $R_t(n(t))$ and N is the total number of dimensions of the query vector $R_t(n(t))$. The Dimension Parameter Weight (DPW) incorporates the observation of user relevance with the value of domains D(n(t)) by providing a better score on the domain values the user has more filled on the query:

$$DPW = \frac{NDT}{N} \tag{12}$$

where NDT is the number of dimensions with the same domain (word, number or price) on the query $R_t(n(t))$ and N is the total number of dimensions of the query vector $R_t(n(t))$. We assign this final Result Score value (RS) to each M-tuple v_o of the answer set A. This value is used by our ISA to reorder the answer set A of Y M-tuples, showing to the user the first set of Z results which have the higher potential value.

3.3 User Iteration

The user, based on the answer set A can now act as an intelligent critic and select a subset of P relevant results, C_P, of A. C_P is a set that consists of P M-tuples $C_P = \{v_1, v_2 \ldots v_P\}$. We consider v_P as a vector of M dimensions; $v_p = (l_{p1}, l_{p2} \ldots l_{pM})$ where l_p are the M different attributes for p = 1, 2..P. Similarly, the user can also select a subset of Q irrelevant results, C_Q of A, $C_Q = \{v_1, v_2 \ldots v_Q\}$. We consider v_q as a vector of M dimensions; $v_q = (l_{q1}, l_{q2} \ldots l_{qM})$ where lq are the M different attributes for q = 1, 2..Q. Based on the user iteration, our Intelligent Search Assistant provides to the user with a different answer set A of Z M-tuples reordered to MD, the minimum distance to the Relevant Centre for the results selected, following the formula:

$$RCP[n] = \frac{\sum_{p=1}^{P} SD_p[n]}{P} = \frac{\sum_{p=1}^{P} l_{pn}}{P} \tag{13}$$

where P is the number of relevant results selected, n the attribute index from the query $R_t(n(t))$ and $SD_p[n]$ the associated Score Dimension vector to the result or M-tuple v_P formed of l_{pn} attributes. An equivalent equation applies to the calculation of the Irrelevant Centre Point. Our Intelligent Search Assistant reorders the retrieved Y set of M-tuples showing only to the user the first Z set of M-tuples based on the lowest distance (MD) between the difference of their distances to both Relevant Centre Point (RD) and the Irrelevant Centre Point (ID) respectively:

$$MD = RD - ID \tag{14}$$

where MD is the result distance, RD is the Relevant Distance and ID is the Irrelevant Distance. The Relevant Distance (RD) of each result or M-tuple v_q is formulated as below:

$$RD = \sqrt{\sum_{n=1}^{N} (SD[n] - RCP[n])^2} \tag{15}$$

where SD[n] is the Score Dimension vector of the result or M-tuple v_q and RCP[n] is the coordinate of the Relevant Centre Point. Equivalent equation applies to the calculation of the Irrelevant Distance. Therefore we are presenting an iterative search progress that learns and adapts to the perceived user relevance based on the dimensions or attributes the user has introduced on the initial query.

3.4 Dimension Learning

The answer set A to the query $R_1(n(1))$ is based on the N dimension query introduced by the user however results are formed of M dimensions therefore the subset of results the user has considered as relevant may have other relevant concepts hidden the user did not considered on the original query. We consider the domain D(m) or the M attributes from which our universe U is formed as the different independent words that form the set of Y results retrieved from the search engines. Our cost function is expanded from the N attributes defined in the query $R_1(n(1))$ to the M attributes that form the searched results. Our Score Dimension vector, SD[m], is now based on M-dimensions. An analogue attribute expansion is applied to the Relevance Centre Calculation, RCP[m]. The query $R_1(n(1))$ is based on the N-Dimension vector introduced by the user however the answer set A consist of Y M-tuples. The user, based on the presented set A, selects a subset of P relevant results, C_P and a subset of Q irrelevant results, C_Q.

Lets consider C_P as a set that consists of P M-tuples $C_P = \{v_1, v_2 \ldots v_P\}$ where v_P is a vector of M dimensions; $v_P = (l_{p1}, l_{p2} \ldots l_{pM})$ and l_p are the M different attributes for p = 1, 2..P. The M-dimension vector Dimension Average, DA[m], is the average value of the m-attributes for the selected relevant P results:

$$DA[m] = \frac{\sum_{p=1}^{P} SD_p[m]}{P} = \frac{\sum_{p=1}^{P} l_{pm}}{P} \tag{16}$$

where P is the number of relevant results selected, m the attribute index of the relation U and $SD_p[m]$ the associated Score Dimension vector to the result or M-tuple v_P formed of l_{pm} attributes. We define ADV as the Average Dimension Value of the M-dimension vector DA[m]:

$$ADV = \frac{\sum_{m=1}^{M} DA[m]}{M} \tag{17}$$

where M is the total number of attributes that form the relation U. The correlation vector σ[m] is the difference between the dimension values of each result with the average vector:

$$\sigma[m] = \frac{\sum_{p=1}^{P} \left(SD_p[m] - DA[m]\right)}{P} = \frac{\sum_{p=1}^{P} \left(1_{pm} - DA[m]\right)}{P} \tag{18}$$

where P is the number of relevant results selected, m the attribute index of the relation U and $SD_p[m]$ the associated Score Dimension vector to the result or M-tuple v_P formed of l_{pm} attributes. We define C as the average correlation value of the M-dimensions of the vector σ[m]:

$$C = \frac{\sum_{m=1}^{M} \sigma[m]}{M} \tag{19}$$

where M is the total number of attributes that form the relation U. We consider an m-attribute relevant if its associated Dimension Average value DA[m] is larger than the average dimension ADV and its correlation value σ[m] is lesser than the average correlation C. We have therefore changed the relevant attributes of the searched entities or ideas by correlating the error value of its concepts or properties represented as attributes or dimensions. On the next iteration, the query $R_2(n(2))$ is formed by the attributes our ISA has considered relevant. The answer to the query $R_2(n(2))$ is a different set A of Y M-tuples. This process iterates until there are not new relevant results to be shown to the user.

3.5 Gradient Descent Learning

Gradient Descent learning is based on the adaptation to the perceived user interests or understanding of meaning by correlating the attribute values of each result to extract similar meanings and cancel superfluous ones. The ISA Gradient Descent learning algorithm is based on a recurrent model. The inputs $i = \{i_1,\ldots,i_P\}$ are the M-tuples v_P corresponding to the selected relevant result subset C_P and the desired outputs $y = \{y_1, \ldots,y_P\}$ are the same values as the input. Our ISA then obtains the learned random neural network weights, calculates the relevant dimensions and finally reorders the results according to the minimum distance to the new Relevant Centre Point focused on the relevant dimensions.

3.6 Reinforcement Learning

The external interaction with the environment is provided when the user selects the relevant result set C_P. Reinforcement Learning adapts to the perceived user relevance by incrementing the value of relevant dimensions and reducing it for the irrelevant ones. Reinforcement Learning modifies the values of the m attributes of the results, accentuating hidden relevant meanings and lowering irrelevant properties. We associate the Random Neural Network weights to the answer set A; W = A. Our ISA updates the network weights W by rewarding the result relevant attributes by:

$$w(p,m) = l_{pm}^{s-1} + l_{pm}^{s-1} * \left(\frac{l_{pm}^{s-1}}{\sum_{m=1}^{M} l_{pm}^{s-1}} \right) \tag{20}$$

where p is the result or M-tuple v_P formed of l_{pm} attributes, m the result attribute index, M the total number of attributes and s the iteration number. ISA also updates the network weights by punishing the result irrelevant attributes by:

$$w(p,m) = l_{pm}^{s-1} - l_{pm}^{s-1} * \left(\frac{l_{pm}^{s-1}}{\sum_{m=1}^{M} l_{pm}^{s-1}} \right) \tag{21}$$

where p is the result or M-tuple v_P formed of l_{pm} attributes, m the result attribute index, M the total number of attributes and s the iteration number. Our ISA then recalculates the potential of each of the result based on the updated network weights and reorders them, showing to the user the results which have a higher potential or score.

4 Validation

The Intelligent Internet Search Assistant we have proposed emulates how Web search engines work by using a very similar interface to introduce and display information. We validate our ISA algorithm with a set of three different experiments. Users in the experiments can both choose between the different Web search engines and the N number of results they would to retrieve from each one. We propose the following formula to measure Web search quality; it is based on the concept that a better search engine provides with a list of more relevant results on top positions. In an list of N results, we score N to the first result and 1 to the last result, the value of the quality proposed is then the summation of the position score based of each of the selected results. Our definition of Quality, Q, can be defined as:

$$Q = \sum_{i=1}^{Y} RSE_i \tag{22}$$

where RSE_i is the rank of the result i in a particular search engine with a value of N if the result is in the first position and 1 if the result is the last one. Y is the total number of results selected by the user. The best Web search engine would have the largest Quality value. We define normalized quality, $\overline{Q}$, as the division of the quality, Q, by the optimum figure which it is when the user consider relevant all the results provided by the Web search engine. On this situation Y and N have the same value:

$$\overline{Q} = \frac{Q}{\frac{N(N+1)}{2}} \tag{23}$$

We define I as the quality improvement between a Web search engine and a reference:

$$I = \frac{QW - QR}{QR} \tag{24}$$

where I is the Improvement, QW is the quality of the Web search engine and QR is the quality reference; we use the Quality of Google as QR in our validation exercise.

4.1 ISA Web Search Engine

In our first experiment, validators can select from which Web search engine they would their results to be retrieved from; as in our first experiment, the users need to select the relevant results. Our ISA combines the results retrieved from the different Web search engines selected. We present the average values for the 18 different queries. We show the normalized quality of each Web search engine selected including our ISA; because users can choose any Web search engine; we are not introducing the improvement value as we do not have a unique reference Web search engine (Table 1).

Table 1. Web search engine validation

Experiment 1–18 queries						
Web	Google	Yahoo	Ask	Lycos	Bing	ISA
$\overline{Q}$	0.2691	0.2587	0.3454	0.3533	0.3429	0.4448

where Web term represents the Web Search Engines selected by the user and Q is the average Quality for the 18 different queries for each Web Search Engine including our ISA.

4.2 ISA Relevant Center Point

In our second experiment we have asked to our validators to search for different queries using only Google; ISA provides with a set of reordered results from which the user needs to select the relevant results. We show the average values for the 20 different queries, the average number of results retrieved by Google and the average number of results selected by the user. We represent the normalized quality of Google and ISA with the improvement of our algorithm against Google. In our third experiment, ISA provides with a reordered list from where the user needs to select which results are relevant. Our ISA reorders the results using the dimension relevant centre point providing to the user with another reordered result list from where the user needs to select the relevant ones. We show the average values for the 16 different queries, the average number of results selected by the user and the average number of results selected. We also represent the normalized quality of Google, ISA and the ISA with the relevant circle iteration including the improvement against Google in both scenarios (Table 2).

Table 2. Relevant center point validation

Experiment 2–20 queries						
Results retrieved	Results selected	Google Q	ISA Q	ISA I	ISA Circle Q	ISA Circle I
19.35	8.05	0.4626	0.4878	15.39 %	–	–
Experiment 3–16 queries						
Results retrieved	Results selected	Google Q	ISA Q	ISA I	ISA Circle Q	ISA Circle I
21.75	8.75	0.4451	0.4595	18 %	0.4953	26 %

where Experiment 2 and 3 results retrieved are the average results shown to the user, results selected are the average results the user considers relevant. Google and ISA Q are the average Quality values based on their different result list ranking. ISA I is the average improvement of our algorithm against Google. ISA Circle Q and I is the average Quality value with its associated Improvement after the first iteration where the user selects the relevant results and our algorithm reorder the results based on the minimum distance to the Relevant Centre Point.

4.3 ISA Learning

Users in this validation can choose between Google and Bing with either Gradient Descent or Reinforcement Learning type. Our ISA then collects the first 50 results from the Web search engine selected, reorders them according to its cost function and finally show to the user the first 20 results. We consider 50 results is a good approximation of search depth as more results can add clutter and irrelevance; 20 results is the average number of results read by a user before he launches another search if he does not find any relevant one.

Our ISA reorders results while learning on the two step iterative process showing only the best 20 results to the user. We present the average Quality values of the Web search engine and ISA for the 29 different queries searched by different users, the learning type and the Web search engine used (Table 3).

Table 3. Learning validation

Gradient descent learning: 17 queries								
First iteration			Second iteration			Third iteration		
Web	ISA	I	Web	ISA	I	Web	ISA	I
0.41	0.58	43 %	0.45	0.61	14 %	0.46	0.62	8 %
Reinforcement learning: 12 queries								
First iteration			Second iteration			Third iteration		
Web	ISA	I	Web	ISA	I	Web	ISA	I
0.42	0.57	34 %	0.47	0.67	36 %	0.49	0.68	0.0 %

where Web and ISA represent the Quality of the selected Web Search Engine and ISA respectively in the three successive learning iterations. The first I represents the improvement from ISA against the Web search; the second I is between ISA iterations 2 and 1 and finally the third I is between the ISA iterations 3 and 2.

5 Conclusions

We have defined a different process; the application of the Random Neural Network as a biological inspired algorithm to measure both user relevance and result ranking based on a predetermined cost function. We have proposed a novel approach to Web search

where the user iteratively trains the neural network while looking for relevant results. Our Intelligent Search Assistant performs generally slightly better than Google and other Web search engines however, this evaluation may be biased because users tend to concentrate on the first results provided which were the ones we showed in our algorithm. Our ISA adapts and learns from user previous relevance measurements increasing significantly its quality and improvement within the first iteration. Reinforcement Learning algorithm performs better than Gradient Descent. Although Gradient Descent provides a better quality on the first iteration; Reinforcement Learning outperforms on the second one due its higher learning rate. Both of them have a residual learning on their third iteration. Gradient Descent would have been the preferred learning algorithm if only one iteration is required; however Reinforcement Learning would have been a better option in the case of two iterations. It is not recommended three iterations because learning is only residual. Deep learning may also be used [19]. Further work includes the validation of our Intelligent Search Assistant with more queries against other search engines such as metasearch engines, online academic databases and recommender systems. This validation comprises its ranking algorithm and its learning performance.

References

1. Wang, X., Zhang, L.: Search engine optimization based on algorithm of BP neural networks. In: Proceedings of the Seventh International Conference on Computational Intelligence and Security, pp. 390–394 (2011)
2. Boyan, J., Freitag, D., Joachims, T.: A machine learning architecture for optimizing Web search engines. In: Proceedings of the AAAI Workshop on Internet-Based Information Systems (1996)
3. Shu, B., Kak, S.: A neural network-based intelligent metasearch engine. Inf. Sci. Inform. Comput. Sci. **120**, 1–11 (2009)
4. Burgues, C., Shaked, T., Renshaw, E., Lazier, L., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proceedings of the 22nd International Conference on Machine Learning, ICML 2005, pp. 89–96 (2005)

5. Bermejo, S., Dalmau, J.: Web metasearch using unsupervised neural networks. In: Proceedings of the 7th International Work-Conference on Artificial and Natural Neural Networks: Part II: Artificial Neural Nets Problem Solving Methods, IWANN 2003, pp. 711–718 (2003)
6. Scarselli, F., Liang, S., Hagenbuchner, M., Chung, A.: Adaptive page ranking with neural networks. In: Proceedings of Special Interest Tracks and Posters of the 14th International Conference on World Wide Web, WWW 2005, pp. 936–937 (2005)
7. Gelenbe, E.: Random neural network with negative and positive signals and product form solution. Neural Comput. **1**, 502–510 (1989)
8. Gelenbe, E.: Learning in the recurrent Random Neural Network. Neural Comput. **5**, 154–164 (1993)
9. Gelenbe, E., Timotheou, S.: Random neural networks with synchronized interactions. Neural Comput. **20**(9), 2308–2324 (2008)
10. Gelenbe, E.: Steps toward self-aware networks. Commun. ACM **52**(7), 66–75 (2009)
11. Gelenbe, E., Wu, F.J.: Large scale simulation for human evacuation and rescue. Comput. Math Appl. **64**(12), 3869–3880 (2012)
12. Filippoupolitis, A., Hey, L., Loukas, G., Gelenbe, E., Timotheou, S.: Emergency response simulation using wireless sensor networks. In: Proceedings of the 1st International Conference on Ambient Media and Systems, vol. 21 (2008)
13. Gelenbe, E., Koçak, T.: Area-based results for mine detection. IEEE Trans. Geosci. Remote Sens. **38**(1), 12–24 (2000)
14. Cramer, C., Gelenbe, E., Bakircloglu, H.: Low bit-rate video compression with neural networks and temporal subsampling. Proc. IEEE **84**(10), 1529–1543 (1996)
15. Atalay, V., Gelenbe, E., Yalabik, N.: The random neural network model for texture generation. Int. J. Pattern Recognit Artif Intell. **6**(1), 131–141 (1992)
16. Gelenbe, E.: Search in unknown random environments. Phys. Rev. E **82**, 061112 (2010)
17. Abdelrahman, O.H., Gelenbe, E.: Time and energy in team-based search. Phys. Rev. E **87** (3), 032125 (2013)
18. Gelenbe, E., Abdelrahman, O.H.: Search in the universe of big networks and data. IEEE Netw. **28**(4), 20–25 (2014)
19. Gelenbe, E., Yin, Y.: Deep learning with random neural networks, paper number 16502. In: International Joint Conference on Neural Networks (IJCNN 2016), World Congress on Computational Intelligence, Vancouver, BC. IEEE Xplore (2016)