A Word Embedding Based Approach for Focused Web Crawling Using the Recurrent Neural Network

P. R. Joe Dhanith[1]*, B. Surendiran[1], S. P. Raja[2]

[1] Department of CSE, National Institute of Technology Puducherry, Karaikal (India)
[2] Department of CSE, Vel Tech Rangarajan Dr.Sagunthala R & D Institute of Science and Technology (India)

Received 11 March 2020 | Accepted 4 July 2020 | Published 25 September 2020

Abstract

Learning-based focused crawlers download relevant uniform resource locators (URLs) from the web for a specific topic. Several studies have used the term frequency-inverse document frequency (TF-IDF) weighted cosine vector as an input feature vector for learning algorithms. TF-IDF-based crawlers calculate the relevance of a web page only if a topic word co-occurs on the said page, failing which it is considered irrelevant. Similarity is not considered even if a synonym of a term co-occurs on a web page. To resolve this challenge, this paper proposes a new methodology that integrates the Adagrad-optimized Skip Gram Negative Sampling (A-SGNS)-based word embedding and the Recurrent Neural Network (RNN).The cosine similarity is calculated from the word embedding matrix to form a feature vector that is given as an input to the RNN to predict the relevance of the website. The performance of the proposed method is evaluated using the harvest rate (hr) and irrelevance ratio (ir). The proposed methodology outperforms existing methodologies with an average harvest rate of 0.42 and irrelevance ratio of 0.58.

Keywords

Focused Crawler, Semantic Similarity, Word Embedding, Adagrad, Cosine, Recurrent Neural Network.

DOI: 10.9781/ijimai.2020.09.003

I. Introduction

There has been a rapid increase in the number of web pages, from only 20 million in 2010 to 1.7 billion in 2020 [1]. The exponential increase in the volume of web pages each year has made it difficult for search engines to index them [2]–[5]. At the heart of a search engine is a web crawler, which is a software bot that retrieves web pages, commencing from seed URLs. A classic web crawler retrieves huge masses of information from the internet for a search, including information on irrelevant topics. Classic web crawlers demand huge storage capacities as well as additional downloading time. Such a problem calls for a topic-driven focused crawler that only downloads relevant web pages from the internet for a given topic.

Fig. 1 illustrates the working design of a focused web crawler, wherein the initial URLs are set by users for a given topic. The crawler visits web pages from initially-defined URLs and computes the similarity score of unexplored web pages. Based on the relevance score, precedence is assigned and stored in a web page archive.

Most focused web crawlers [6]–[9] only use full-page text to compute the similarity score of a web page, while others [10]–[13] use both full-page and anchor texts to calculate the relevance score, and [14]–[16] use cosine similarity to calculate the similarity score of unvisited web pages. In numerous existing studies [14]–[16], the cosine similarity value is calculated by finding the TF-IDF. The TF-IDF-based cosine similarity calculates the relevance score only if the topic term co-occurs with the terms on the web page, or else the similarity value is set to zero. The cosine similarity-based focused crawler provides a zero similarity score if the web page is semantically related but with no terms in common between the web page and the topic.

It was to overcome such challenges that researchers began working on ontology learning-based crawlers [17]–[19] to establish the semantic similarity between the topic and web page. Domain-specific ontologies are designed by domain experts, and crawlers fetch wrong results when human errors or discrepancies occur in the ontologies.

This paper proposes a new word embedding-based approach using the RNN to resolve this issue. Word embedding is one of the most common web page vocabulary representations. It is capable of capturing the meaning of a word on a web page, its semantic and syntactic similitude, and its relationship with other words. This work uses the A-SGNS to handle rare words that show up in the vocabulary. From the embedding matrix, the cosine similarity between the topic and web pages is calculated. The calculated cosine vectors are given as input to the RNN to predict the relevance of the web page.

The major contributions of this paper are as follows:

(1) Integrating the A-SGNS model with the RNN to automatically retrieve relevant web pages,

(2) Optimizing the SGNS using the Adagrad algorithm and the RNN using the RMSprop algorithm, and

(3) Implementing and evaluating the following nine different focused crawlers, including the breadth-first search (BFS)-based, vector space model (VSM), ontology learning-based using artificial neural network (ANN), Naive Bayes (NB)-based, link context-based, ANN-

* Corresponding author.
E-mail address: joe.dhanith@gmail.com

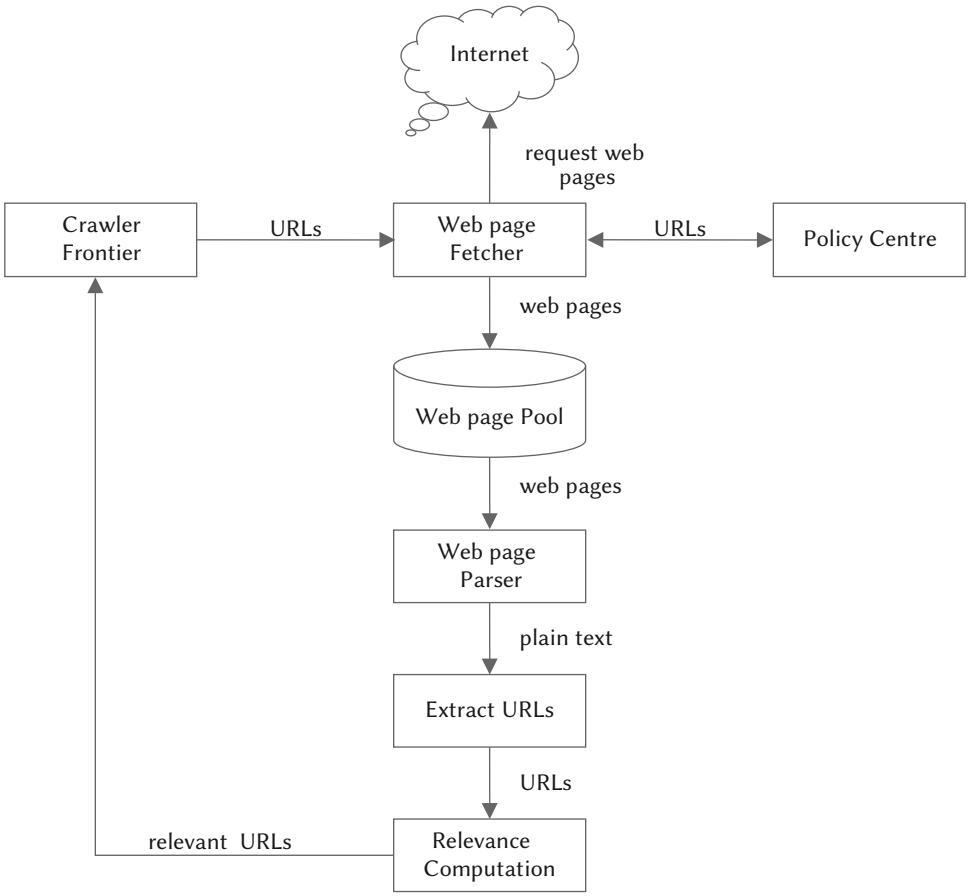

Fig. 1. Working design of focused web crawler.

based, semi-supervised, optimized Naive Bayes (ONB)-based, and the proposed RNN crawler. The efficiency of the nine focused crawlers is assessed using the harvest rate and irrelevance ratio.

The remainder of this article is organized as follows: Section II addresses existing methodologies. Section III describes the newly-constructed crawler framework, and Section IV the experimental design. Section V presents the experimental analysis, and Section VI concludes the paper.

A. Working of Focused Web Crawler

Fig. 1 shows the working architecture of the focused web crawler, whose major components are a crawler frontier, web page fetcher, policy centre, web page pool, web page parser and relevance computation. The crawler frontier is a priority queue that stores a list of URLs in a prioritized order, based on the relevance score. The web page fetcher downloads web pages. The policy centre checks whether the web page is downloadable, and the web page pool stores downloaded web pages. The web page parser parses the web page to plaintext. The relevance computation computes the relevance of unvisited web pages. The stepwise working of the focused web crawler is as follows:

Step 1:The crawler frontier is initialized with the seed URLs and the policy centre with the depth of the web pages explored.

Step 2: The web page fetcher downloads web pages in the crawler frontier one by one. Once a web page is downloaded, the web page fetcher extracts the URLs present therein and sends them to the policy centre.

Step 3: The policy centre checks the downloadability of the received URL. A downloadable URL is sent back to the web page fetcher for downloading, else it is terminated. Steps (2) and (3) are repeated until the user-defined depth is reached.

Step 4: Once the web page fetcher receives the URL from the policy centre, it downloads the web page and stores it in the web page pool as a HTML document.

Step 5: The stored web pages are sent to the web page parser, along with the URLs, to retrieve meaningful information.

Step 6: The extracted information snippets are despatched to the relevance computation module to determine the relevance of the web page to the given topic. If the web page is relevant, the extracted URL is sent to the crawler frontier or else it is terminated.

II. Related Work

The Google search engine uses the PageRank algorithm [2] to compute the relevance score of web pages. The PageRank algorithm is a voting method based on the number of incoming links and the rank of incoming links. If the number and the rank of the incoming links are high, the PageRank of the web page is also correspondingly high. The Google PageRank algorithm is formulated as follows in Equation (1),

$$p(w_p) = (1 - d_f) + d_f \cdot \left(\frac{p(l_1)}{c(l_1)} + \frac{p(l_2)}{c(l_2)} + \cdots + \frac{p(l_n)}{c(l_n)}\right) \quad (1)$$

where d_f is the damping factor, the value of d_f is usually set to 0.85, $p(w_p)$ is the PageRank of the web page (w_p), $l_1, l_2, \ldots, l_n$ are the incoming links to the web page w_p, $p(l_1)$ is the PageRank of the first incoming link (l_1), $c(l_1)$ is the number of outgoing links from web page l_1.

The baseline focused crawler downloads only relevant web pages by computing the relevance score of target variables such as full-page terms and anchor terms. The priority of the unvisited hyperlinks is calculated by combining the relevance score of the target variables. The priority score is given as a cosine function, as shown in Equation (2),

$$f_p(url) = \frac{f_{rs}(t, p) + f_{rs}(t, a)}{2} \quad (2)$$

where $f_p(url)$ is the priority function of unvisited URL, $f_{rs}(t, p)$ is the cosine function between the given topic and the full page terms, $f_{rs}(t, a)$ is the cosine function between the given topic and the anchor terms.

Andrea Capuano et al. [20] designed an ontology learning-based focused crawler using the convolution neural network (CNN). This work uses the Dbpedia spotlight and ImageNet to annotate, respectively, web page text and image data. A Li [21] semantic similarity algorithm calculates the textual relevance between the topic and the web page, and a CNN algorithm computes the relevance score between the downloaded image and the image in the knowledge base. The classification of the text and image is combined to identify the relevance of the web page. This work produced an average harvest rate of 0.29 after 5000 web page downloads.

Javad Hosseinkhani et al. [22] proposed an ontology learning-based focused crawler using the ant colony optimization (ACO) algorithm. The crime ontology builder in this work is used to design a crime ontology repository that annotates web pages. The ACO crawls and prioritizes web pages from the internet, while the support vector machine (SVM) classifies the relevance of a particular web page.

Debajyoti Mukhopadhyay et al. [23] advanced a semantic focused crawler to download relevant URLs. This work proposed a relevance score calculation formula between the topic and the web page, as shown in Equation (3),

$$f_{rs} = \sum f_o . N_o + \sum f_s . N_s \quad (3)$$

where f_{rs} is the relevance score function, f_o is the ontology-based term weight, N_o is the count of the terms in the ontology, f_s is the synonym weight value of the term, and N_s is the count of the synonyms in the ontology. A web page with a relevance score above 0.5 is considered relevant, otherwise it is not.

Juan Qiu et al. [24] designed a focused crawler for the OpenStack Questions and Answers (Q&A) knowledge base. This work uses the linear discriminant analysis (LDA) clustering algorithm to construct the QA knowledge base topic corpus. A VSM is applied to find the similarity between the topic and the web page for corpus update.

Tanaphol SUEBCHUA et al. [25] propounded a history feature-based focused crawler. The history feature is extracted to reduce the priority score of the unvisited web page that downloads irrelevant web pages consecutively. The history feature, along with the relevance score of the link context and page text, is given as input to a NB classifier to predict the relevance of the web page.

Guangxia Xu et al. [26] elucidated a focused crawler based on particle swarm optimization (PSO). Initially, the TF-IDF is applied to calculate the weight of the terms, following which the PSO is applied to predict the relevance of the web page.

H.Dong et al. [27] discussed a self-adaptive semantic focused (SASF) crawler that combines an information content (IC)-based semantic similarity measure and a statistics-based similarity measure to determine the similarity score of a web page with respect to a given topic. The relevance score is calculated only for a full-page text feature with the given topic.

The relevance score of the SASF [27] is computed using Equation (4).

$$f_{rs}(url) = \max\left(f_{ic}(t, p), f_{stsm}(t, p)\right) \quad (4)$$

where $f_{rs}(url)$ relevance score function of the URL, $f_{ic}(t, p)$ is the Information content based semantic similarity function between the given topic and the full page terms, $f_{stsm}(t, p)$ is the statistical based string matching between the given term and the full page terms.

Ya Jun et al. [28] designed a cell-like membrane computing optimization (CMCFC) algorithm. The relevance score is calculated for four target variables (web page contents, link context, title term and the surrounding paragraph text) with the given term, using the cosine-based similarity metric. The relevance score of the four target variables is combined to compute the precedence of the unexplored web pages.

The relevance score of the CMCFC [28] is computed using Equation (5),

$$f_{rs}(url) = f_{rs}(t, p) + f_{rs}(t, a) + f_{rs}(t, title) + f_{rs}(t, st) \quad (5)$$

where $f_{rs}(url)$ is the relevance score function of the URL, $f_{rs}(t, p)$ is the cosine function between the given topic and the full page terms, $f_{rs}(t, a)$ is the cosine function between the given topic and the anchor terms, $f_{rs}(t, title)$ is the cosine function between the given topic and the terms, rs (t, st) is the cosine function between the given topic and the surrounding terms.

Hai-Tao Zheng et al. [19] elucidated a semantic focused crawler based on the artificial neural network (ANN). The relevance score is calculated, based on the distance between the full-page text and the given topic in the ontology. The crawler computes the term frequency of the unvisited web pages and feeds them as input to the ANN. The relevance score of the unvisited web pages is the output of the ANN. A major limitation of this approach is its inability to work well in an uncontrolled web environment.

Ahmed I Saleh et al. [17] designed a focused crawler using the optimized NB classifier (ONB). This work integrates the NB classifier with the SVM to form an optimized NB classifier. An integrated SVM and genetic algorithm is used to remove outliers in the training samples. The training data with the outliers removed is thereafter used to train the NB classifier. The ONB finds the sense of the data using the D^2O ontology, calculates the similarity score of the web page, and determines its relevance.

Hai Dong et al. [18] designed a semi-supervised ontology learning-based approach for focused web crawling. This work extracts the Resnik semantic similarity score [29] and the statistical-based co-occurrence similarity score between the topic and the web page contents as features. The feature vector is then given as input to the SVM classifier to predict the relevance of the web page.

A review of the literature revealed the following drawbacks:

1. The TF-IDF weighting scheme finds the relevance of a web page only if the topic term co-occurs in target variables such as web page text and anchor text. The relevance score is otherwise calculated as zero, given that the TF-IDF does not consider the semantic similarity of the web page.
2. If the number of words on a web page is high, the dimension of the feature space generated by the TF-IDF is also high. The dimensionality of the feature space using the TF-IDF depends on the number of words on the web page. The TF-IDF vectors of web pages cause the high-dimensionality feature space that results in inaccurately-performing NB, SVM and ANN classifiers in a crawling environment.
3. Learning ontological concepts during a dynamic crawling process is an expensive, time-consuming process for basic learning algorithms like the NB, SVM and ANN. The complexity of learning ontological concepts in a crawling environment culminates in existing ontology learning-based crawlers performing at levels below par.

To solve these issues, this paper proposes a new word embedding-based approach using the RNN. An A-SGNS model is used to build a

VSM for representing words through a low-dimensional space. From the derived matrix, the cosine similarity between the extracted topics and the extracted web page terms is calculated to form a feature vector. The generated cosine feature vector is given as an input to the RNN to predict the relevance of the web page.

III. Proposed Methodology

Fig. 2 shows the workflow diagram of the proposed work, with a six-layer architecture. The crawler frontier is initialized with the seed URLs and the topic by the user. The first layer is the topic preprocessing layer, where the given topic is preprocessed by applying methodologies like tokenization, Parts-of-Speech (POS) tagging, nonsense word filtering, stemming and synonym searches. The preprocessing is done using the Python Natural Language Toolkit (NLTK) library [30], [31], and the synonyms of the given topic are extracted for a meaningful search. The preprocessed topic terms are stored in a storage. The second layer is the crawling layer, where web pages are downloaded from the web, starting from manually assigned seed URLs. Once the download is done, the web pages are sent to the term extraction layer. The third layer is the term extraction layer, where the web pages are parsed to plaintext by removing HTML tags. After the parsing, target variables such as web page text and anchor text are extracted from the web page. The fourth layer is the term preprocessing layer, where the extracted target variables are preprocessed by applying methodologies like tokenization, POS tagging, nonsense word filtering and stemming. The preprocessing is carried out using the NLTK library [30], [31]. The fifth layer is the feature extraction layer, where the A-SGNS-based word embedding matrix is formed. From the derived matrix, the cosine similarity between the extracted topics and the extracted terms is calculated to form a feature vector. The generated cosine feature vector is given as input to the classification layer, where the recurrent neural network classifies the web page to determine its relevance.

A. Recurrent Neural Networks

H. Palangi et al., [32]–[34] proposed a RNN for sentence embedding. The RNN, a type of deep learning model, uses the previous output as input in the hidden state and maintains the previous output to predict the current output. Fig. 3 shows the RNN workflow architecture of the proposed work.

The hidden state can be formulated as shown in equation (6):

$$s(t) = \tanh(w_{in}x(t) + w_{rec}s(t-1) + bias_s) \quad (6)$$

where w_{in} is the input weight vector, w_{rec} is the recurrent weight vector, s(t) is the hidden state, x(t) is the input vector, s(t–1) is the previous hidden state and $bias_s$ is the bias.

The output can be formulated as in equation (7):

$$o(t) = \sigma(w_{out}s(t) + bias_{out}) \quad (7)$$

where $\sigma(.)$ is the sigmoid activation function, o(t) is the output vector, w_{out} is the output weight, and $bias_{out}$ is the bias of the output state.

B. Feature Extraction

The first step in this work is to build a VSM to represent words through a low-dimensional space, using prediction-based word embedding. The A-SGNS [35]–[37], a prediction-based model which follows the neural network approach, is used. Given a sample of vocabulary, V, and the retrieved word context pair set, Z, let p(Z=1|(w, c)) be the likelihood that (w, c) arrives from Z and let p(Z=0|(w, c)) be the likelihood that (w, c) may not. The presupposition of SGNS is that if c is the context of word w in a window, the conditional probability of p(Z=1|(w, c))should be high, and otherwise small. Let v_w denote the vector representation of w, and v_c denote the vector of c. D is the set of all pairs (w, c) that are in the text and D' is the set of all pairs (w, c) not in the text. Then, Z can be represented as follows in equation (8).

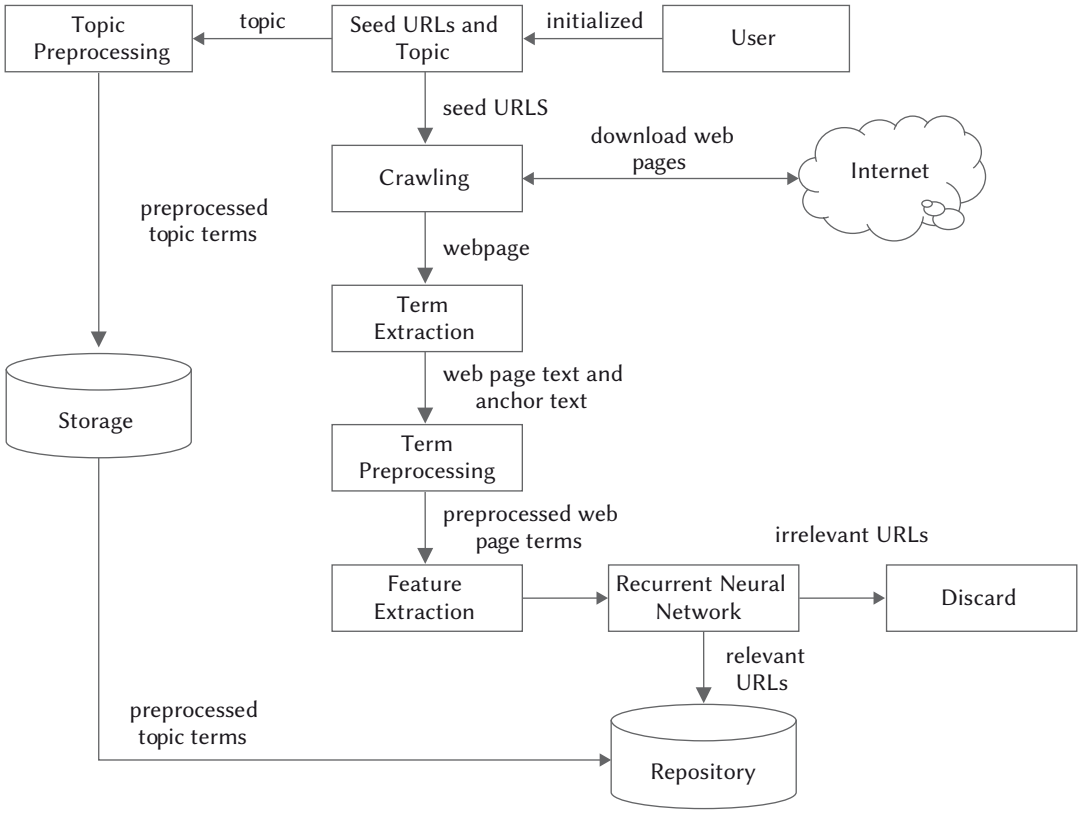

Fig. 2. Proposed workflow architecture.

$$Z = \begin{cases} 1 & if\ (w, c)\ \in D \\ 0 & if\ (w, c)\ \in D' \end{cases} \tag{8}$$

Then the p(Z=1|(w, c)) and p(Z=0|(w, c)) are computed as shown in equations (9) and (12) respectively,

$$p(Z = 1|(w, c)) = \frac{1}{1 + e^{v_c{}^T . v_w}} \tag{9}$$

$$p(Z = 0|(w, c)) = 1 - \frac{1}{1 + e^{v_c{}^T . v_w}} = \frac{e^{v_c{}^T . v_w}}{1 + e^{v_c{}^T . v_w}} \tag{10}$$

the above equation (10) is multiplied using $\frac{e^{-v_c{}^T . v_w}}{e^{-v_c{}^T . v_w}}$ and the following equations (11) and (12) are formulated.

$$p(Z = 0|(w, c)) = \frac{e^{v_c{}^T . v_w}}{1 + e^{v_c{}^T . v_w}} . \frac{e^{-v_c{}^T . v_w}}{e^{-v_c{}^T . v_w}} \tag{11}$$

$$p(Z = 0|(w, c)) = \frac{1}{1 + e^{-v_c{}^T . v_w}} \tag{12}$$

$$p(Z|(w, c); \theta) = \left(\frac{1}{1 + e^{v_c{}^T . v_w}}\right)^Z . \left(\frac{1}{1 + e^{-v_c{}^T . v_w}}\right)^{1-Z} \tag{13}$$

where $\theta=(v_c, v_w)$

$$L(\theta) = \prod_{(v_w, v_c) \in D \cup D'} \left(\frac{1}{1 + e^{v_c{}^T . v_w}}\right)^Z . \left(\frac{1}{1 + e^{-v_c{}^T . v_w}}\right)^{1-Z} \tag{14}$$

log likelihood is applied on both sides in equation (14) and formulated in the following equation (15).

$$l(\theta) = \sum_{(v_w, v_c) \in D \cup D'} Z . \log\left(\frac{1}{1 + e^{v_c{}^T . v_w}}\right) + (1 - Z) . \log\left(\frac{1}{1 + e^{-v_c{}^T . v_w}}\right) \tag{15}$$

$$l(\theta) = \sum_{(v_w, v_c) \in D} \log\left(\frac{1}{1 + e^{v_c{}^T . v_w}}\right) + \sum_{(v_w, v_c) \in D'} \left(\log\left(\frac{1}{1 + e^{-v_c{}^T . v_w}}\right)\right) \tag{16}$$

Let $\sigma(v_c{}^T . v_w) = \frac{1}{1+e^{v_c{}^T . v_w}}$ and $\sigma(-v_c{}^T . v_w) = \frac{1}{1+e^{-v_c{}^T . v_w}}$ then the equation (16) can be formulated as follows in equation (17),

$$l(\theta) = \log\left(\sigma(v_c{}^T . v_w)\right) + \sum_{i=1}^{k} E_{v_w \sim p_n(w)} \left[\log\left(\sigma(-v_c{}^T . v_w)\right)\right] \tag{17}$$

where $p_n(w) = \frac{\#w}{\sum_{w'} \#w'}$

Then the cost function can be given as follows in equation (18):

$$J(\theta; v_w, v_c) = \sum_{v_w \in w} \sum_{v_c \in w} \#(v_w, v_c) . \log\left(\sigma(v_c{}^T . v_w)\right) + \sum_{i=1}^{k} E_{v_w \sim p_n(w)} \left[\log\left(\sigma(-v_c{}^T . v_w)\right)\right] \tag{18}$$

The goal of any machine learning model is to find the optimal values of a weight matrix (θ) to minimize prediction errors. To update the lower learning rates for frequent words and higher learning rates for infrequent words, this work uses the Adagrad algorithm [38]–[40] for optimizing the cost function. The gradient descent on the cost function is applied with respect to θ, as shown in Equation (19):

$$\frac{\partial J}{\partial \theta} = \#(v_w, v_c) . (\sigma(v_c{}^T . v_w) - 1) . \frac{\partial v_c{}^T . v_w}{\partial \theta} + \sum_{i=1}^{k} \sigma(-v_c{}^T . v_w) . \frac{\partial v_c{}^T . v_w}{\partial \theta} \tag{19}$$

A general AdaGrad update equation for cost function can be given in the following equation (20):

$$\theta = \theta - \frac{\omega}{\sqrt{\sum_{\tau=1}^{t-1} \frac{\partial J}{\partial \theta}^2}} \frac{\partial J}{\partial \theta} \tag{20}$$

From the designed word embedding model, the cosine similarity between the given topic and the web page is extracted as a feature. The cosine similarity between the topic and the content of the web page is given as follows in Equation (21):

$$sim(t, d) = \frac{\vec{t}^T . \vec{d}}{\|\vec{t}\| . \|\vec{d}\|} \tag{21}$$

where $\vec{t}^T$ is the embedding vector corresponding to the Topic, $\vec{d}$ is the embedding vector corresponding to the web page contents.

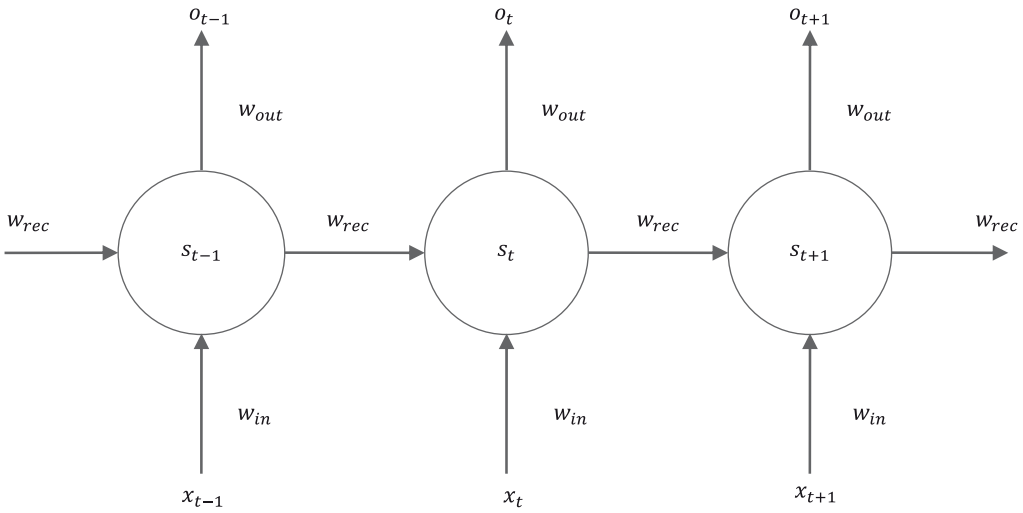

Fig. 3. Recurrent Neural Network workflow architecture of the proposed work.

C. Classification Layer

The embedding vector of a document $\vec{d}$ is represented as $\{\overrightarrow{d_p}, \overrightarrow{d_{n_1}}, \overrightarrow{d_{n_2}}, \ldots, \overrightarrow{d_{n_m}}\}$.

where $\overrightarrow{d_p}$ is the positive sample among the web page documents and $\overrightarrow{d_{n_k}}$ is the kth negative sample of the same. These semantic vectors are produced by feeding the web page documents into the neural network (RNN), as discussed in section IIIA.

1. Recurrent Weight

To maximize the likelihood of the positive document for the given document with respect to recurrent weight (w_{rec}) can be formulated as follows in equation (22):

$$L(w_{rec}) = \min_{w_{rec}} \left\{ -\log \prod_{i=1}^{N} P(\overrightarrow{d_p}|\vec{t}) \right\} \tag{22}$$

where w_{rec} is the recurrent weight, $P(\overrightarrow{d_p}|\vec{t})$ is the probability of positive web page document for the ith Topic, and N is the number of topic-document pair in the corpus.

The above equation can be rewritten as follows in equation (23):

$$L(w_{rec}) = \min_{w_{rec}} \sum_{i=1}^{n} l_i(w_{rec}) \tag{23}$$

The $l_i(w_{rec})$ can be determined using the formula below from (24)-(28):

$$l_i(w_{rec}) = -\log\left(\frac{e^{\gamma.sim(t_i,d_i^+)}}{e^{\gamma.sim(t_i,d_i^+)} + \sum_{j=1}^{n} e^{\gamma.sim(t_i,d_{i,j}^-)}}\right) \tag{24}$$

$$l_i(w_{rec}) = \log\left(\frac{e^{\gamma.sim(t_i,d_i^+)} + \sum_{j=1}^{n} e^{\gamma.sim(t_i,d_{i,j}^-)}}{e^{\gamma.sim(t_i,d_i^+)}}\right) \tag{25}$$

$$l_i(w_{rec}) = \log\left(1 + \sum_{j=1}^{n} e^{\gamma.sim(t_i,d_{i,j}^-)} . e^{-\gamma.sim(t_i,d_i^+)}\right) \tag{26}$$

$$l_i(w_{rec}) = \log\left(1 + \sum_{j=1}^{n} e^{-\gamma.\left(sim(t_i,d_i^+) - sim(t_i,d_{i,j}^-)\right)}\right) \tag{27}$$

$$l_i(w_{rec}) = \log\left(1 + \sum_{j=1}^{n} e^{-\gamma.\Delta_{i,j}}\right) \tag{28}$$

where $\Delta_{i,j} = sim(t_i, d_i^+) - sim(t_i, d_i^-)$, the $\Delta_{i,j}$ value lies between 0 to 1, and γ is a scaling factor to increase the range of $\Delta_{i,j}$.

To perform back propagation through time [41] for $L(w_{rec})$ with respect to recurrent weight (w_{rec}) can be derived as follows in equation (29):

$$\frac{\partial L(w_{rec})}{\partial w_{rec}} = \sum_{i=1}^{n} \frac{\partial l_i(w_{rec})}{\partial w_{rec}} \tag{29}$$

The derived cost value of recurrent weight (w_{rec}) can be given as follows in equation (30):

$$\frac{\partial L(w_{rec})}{\partial w_{rec}} = \sum_{i=1}^{N} \sum_{j=1}^{n} \sum_{t=1}^{T} \alpha_{i,j,t} . \frac{\partial \Delta_{i,j,t}}{\partial w_{rec}} \tag{30}$$

where T is the number of time steps that the network is unfold over time and

$$\alpha_{i,j,t} = \frac{-\gamma . \sum_{j=1}^{n} e^{-\gamma.\Delta_{i,j,t}}}{1 + \sum_{j=1}^{n} e^{-\gamma.\Delta_{i,j,t}}}$$

The recurrent weight can be updated by using the RMSprop algorithm [42] because of its ability to update the lower learning rates for frequent parameters and higher learning rates for infrequent parameters and also clip the gradient when it goes higher than a threshold.

$$E[g^2]_t = \alpha E[g^2]_{t-1} + (1-\alpha)\left(\frac{\partial L(w_{rec})}{\partial w_{rec}}\right)^2 \tag{31}$$

$$w_{rec} = w_{rec} - \frac{\omega}{\sqrt{E[g^2]_t}} \frac{\partial L(w_{rec})}{\partial w_{rec}} \tag{32}$$

where $E[g^2]$ is the mean square of the gradient, α is the moving average parameter which is usually set to 0.9, ω is the learning rate which is set to 0.001, at each time step τ for the parameter w_{rec}.

2. Input Weight

As derived for recurrent weight, the cost value of input weight (w_{in}) can be derived as follows in equation (33):

$$\frac{\partial L(w_{in})}{\partial w_{in}} = \sum_{i=1}^{N} \sum_{j=1}^{n} \sum_{t=1}^{T} \alpha_{i,j,t} . \frac{\partial \Delta_{i,j,t}}{\partial w_{in}} \tag{33}$$

IV. Experimental Design and Analysis

A prototype of the crawlers (BFS, VSM, SVM, NB, ANN, ontology learning-based using the ANN, semi-supervised using the SVM, ONB-based and, finally, the proposed RNN) was developed in Python3 [43], [44], within the Spyder3.6 [45] platform. A cluster of six systems, each with the following configurations, was used to implement the prototypes: (i) 2.20GHz Intel Core i7-8750H 8th Gen processor, (ii) 16GB DDR4 RAM, (iii) 1TB serialATA hard drive, (iv) NVidia GeForce GTX 1060 6GB graphics, and (v) the Windows 10 operating system. These prototypes were implemented to crawl from the real web, using the Python packages, BeautifulSoup [46] and urllib [47]. BeautifulSoup package was used to handle HTML documents and urllib package was used to handle the URLs. The lxml parser [48] of BeautifulSoup package was used to parse the HTML documents. The urllib.parse function was used to parse the URLs. A set of ten topics and their respective seed URLs, as shown in Table 1, were given as input to all the crawlers. We collected 350000 (175000 positive and 175000 negative samples) URLs, along with their web page contents, for the topics shown in Table I in order to train the machine learning algorithms.

The experimental evaluations were carried out in two stages. The first stage was the training-testing phase of the machine learning algorithms, where the NB+TF-IDF, SVM+TF-IDF, ANN + TF-IDF and the proposed RNN + A-SGNS crawlers were evaluated using the metrics in Section V(A). The second stage was the crawling phase, where the performance of the crawlers (BFS, VSM, ontology learning-based using the ANN, NB-based, link context-based using the SVM, ANN-based, semi-supervised using the SVM, optimized Naive Bayes-based and the proposed RNN+A-SGNS) was evaluated using the metrics in Section V(C).

The NB, SVM and ANN algorithms, along with the TF-IDF, were implemented using the sci-kit learn Python package [49]. The NB-based crawler was implemented using the Gaussian Naive Bayes (GNB) classifier with a Laplace smoothing function, and the SVM-based crawler using a degree 1 linear SVM. The ANN model with 4 hidden nodes was implemented using the stochastic gradient descent (SGD) optimizer with the initialized weight value of 0.5 and learning rate of 0.1. In the proposed RNN model, the recurrent weight (w_{rec}) was initialized to -1.5 and the input weight (w_{in}) was initialized to 2.0. The learning rate ω was initialized to 0.001 for both w_{rec} and w_{in}.

TABLE I. Seed URLs for the Ten Topics

S.No	Topic	Seed URL
1	Football	• https://en.wikipedia.org/wiki/Football • https://www.bbc.co.uk/sport/football
2	Knowledge Mapping	• https://www.apqc.org/blog/4-step-guide-knowledge-mapping • https://www.mindmeister.com/blog/build-knowledge-map/
3	Robot Army	• https://en.wikipedia.org/wiki/Military_robot • https://www.popularmechanics.com/technology/robots/a29610393/robot-soldier-boston-dynamics/
4	Smart Phone	• https://en.wikipedia.org/wiki/Smartphone • https://www.amazon.in/Smartphones/b?ie=UTF8&node=1805560031
5	Cloud Computing	• https://en.wikipedia.org/wiki/Cloud_computing • https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/
6	wildfires	• https://en.wikipedia.org/wiki/Wildfire • https://simple.wikipedia.org/wiki/Wildfire
7	Shahrukh khan	• https://en.wikipedia.org/wiki/Shah_Rukh_Khan • https://www.imdb.com/name/nm0451321/
8	computer	• https://en.wikipedia.org/wiki/Computer • https://www.webopedia.com/TERM/C/computer.html
9	Apple	• https://www.apple.com/in/ • https://minecraft.gamepedia.com/Apple
10	Movie	• https://www.amctheatres.com/movies • https://www.imdb.com/chart/moviemeter/

V. Performance Evaluation

A. Performance Evaluation of Training Phase

1. Performance Metrics

This work uses four different metrics to measure the efficiency, at the training phase of different machine learning algorithms. They are accuracy (a), precision (p), recall (r) and F1-score (f) as shown in the following Equations (34), (35), (36), and (37) respectively.

$$a = \frac{tp + tn}{tp + tn + fp + fn} \tag{34}$$

$$p = \frac{tp}{tp + fp} \tag{35}$$

$$r = \frac{tp}{tp + fn} \tag{36}$$

$$f = \frac{2 * p * r}{p + r} \tag{37}$$

where *tp*, *tn*, *fp* and *fn* are true positive, true negative, false positive and false negative respectively.

B. Analysis of Training Phase

A series of experiments was conducted to identify the right classifier with the requisite ability to guide the focused crawler. A dataset with 350,000 positive query-document pairs was collected for 10 different topics, as shown in Table I, each with 17,500 positive and 17,500 negative samples. Initially we applied tokenization, POS tagging, nonsense word filtering and stemming on both query and document data. The preprocessing was carried out using the Python Natural Language Toolkit (NLTK) [30], [31]. The nltk.word_tokenize() function was used to tokenize the topic words and the document words, the nltk.pos_tag() function to find the part of speech of each topic word and document word, and the nltk.stem package to find the root word of each topic word and document word. The words identified without POS tag were removed as non-sense words. Following the preprocessing of the training data, the TF-IDF-based cosine similarity and A-SGNS-based cosine similarity were extracted as a feature for each query-document pair. The TF-IDF-based extracted feature was used to train the NB, SVM, and ANN classifiers, while the A-SGNS-based extracted feature was used to train the RNN classifier. After training the classifiers, a testing dataset of 2827 query-document pairs was used to test the performance of the classifiers. The training phase was evaluated using four well-known metrics, formulated in Equations (34)-(37). Table II shows the results of a comparison of the four classifiers with 350,000 training data samples. The SVM with the TF-IDF, NB with the TF-IDF, ANN with the TF-IDF, and RNN with the A-SGNS produced accuracy of 0.623, 0.62, 0.70 and 0.813, respectively.

Logistic regression works well with linear data but not so with non-linear data. To predict categorical outcomes, it needs each data point to be independent. Given the limitations involved, it was, consequently, unable to perform well on the dynamic internet. Since the number of words in the web page was high, the dimensions created by the TF-IDF vectors were also high. In a high-dimensional feature space, the NB, SVM and ANN were affected by problems with overfitting and time consumption [50]. The NB, SVM and ANN failed to handle high-dimensional feature vectors and produced inaccurate results. The RNN, on the other hand, is a discriminative model that tries to differentiate between positive and negative samples in order to undertake the classification. In the proposed work, the A-SGNS model was used to build a VSM to represent words through a low-dimensional space. The ability of the RNN to handle the A-SGNS word embedding vectors resulted in its enhanced performance in a dynamic web environment [51], with an average accuracy of 0.813.

TABLE II. Precision, Recall, F1-score and Accuracy with 350,000 Training Samples

Algorithm	Precision		Recall		F1-score		Accuracy
	Class 1	Class 0	Class 1	Class 0	Class 1	Class 0	
SVM + TF-IDF	0.50	0.51	0.62	0.39	0.55	0.44	0.623
NB + TF-IDF	0.50	0.513	0.626	0.39	0.55	0.443	0.62
ANN + TF-IDF	0.5	0.50	0.42	0.58	0.45	0.53	0.70
RNN + A-SGNS	0.62	0.57	0.45	0.73	0.52	0.64	0.813

C. Performance Evaluation of Crawling Phase

1. Performance Metrics

The performance of the six focused crawlers were measured by using harvest rate and irrelevance ratio can be shown in the equations (38) and (39).

2. Harvest Rate

Harvest rate is defined as the ratio of the number of relevant web pages downloaded out of total number of web pages downloaded. The harvest rate *(hr)* can be formulated as follows in equation (38).

$$hr = \frac{R_{wp}}{N_{wp}} \tag{38}$$

where hr is the harvest Rate, R_{wp} is the number relevant web pages downloaded, and N_{wp} is the total number of web pages downloaded.

3. Irrelevance Ratio

Irrelevance ratio is defined as the ratio of number of irrelevant web pages downloaded out of total number of web pages downloaded. The irrelevance ratio can be formulated as follows in equation (39).

$$ir = \frac{r_i \cap n_i}{n_i} \quad (39)$$

where ir is the irrelevance ratio, r_j is the number of relevant web pages downloaded, and n_j is the total number of web pages downloaded.

D. Analysis of Crawling Phase

The experimental results were evaluated for all the four focused crawlers, namely, the SVM + TF-IDF, NB + TF-IDF, ANN + TF-IDF and the proposed RNN + A-SGNS. For the NB, SVM, and ANN, the TF-IDF-based cosine similarity was given as an input feature, while for the RNN, the SGNS-based cosine similarity was the input feature.

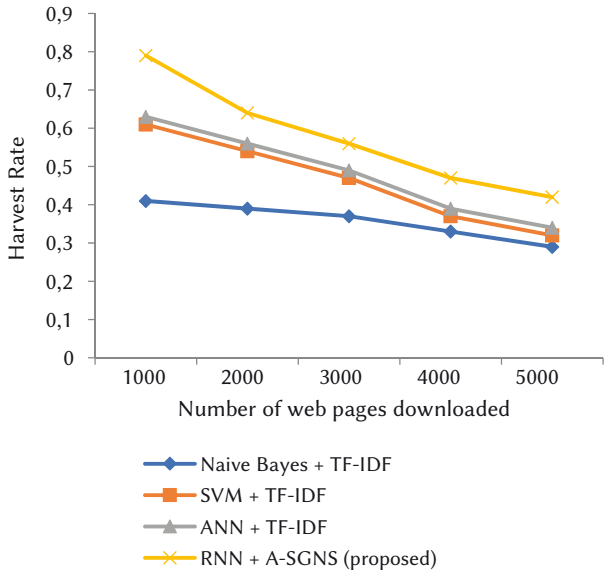

Fig. 4. Average harvest rate for ten topics for the SVM + TF-IDF, NB + TF-IDF, ANN + TF-IDF and RNN + A-SGNS crawlers.

Fig. 4 shows the average harvest rate and Fig. 5 shows the average irrelevance ratio of the SVM + TF-IDF, NB + TF-IDF, ANN + TF-IDF and RNN + SGNS crawlers, respectively. The TF-IDF-based features consider similarity only if the topic term co-occurs on the web page. As a result, the SVM + TF-IDF, NB + TF-IDF crawler, and ANN + TF-IDF crawler considers most web pages that are semantically related to the topic as irrelevant. The SVM+TF-IDF, NB + TF-IDF and ANN + TF-IDF crawlers produced an average harvest rate of 0.32, 029 and 0.34, along with a high irrelevance ratio of 0.68, 0.71 and 0.66,respectively. The A-SGNS is a context learning-based algorithm that considers the semantic relatedness between the topic and the web page term. Owing to this advantage, it considers the semantically related web page as a relevant web page, and produced an average harvest rate of 0.42 and a low irrelevance ratio of 0.58, thus outperforming the other focused SVM + TF-IDF, NB+ TF-IDF and ANN + TF-IDF crawlers.

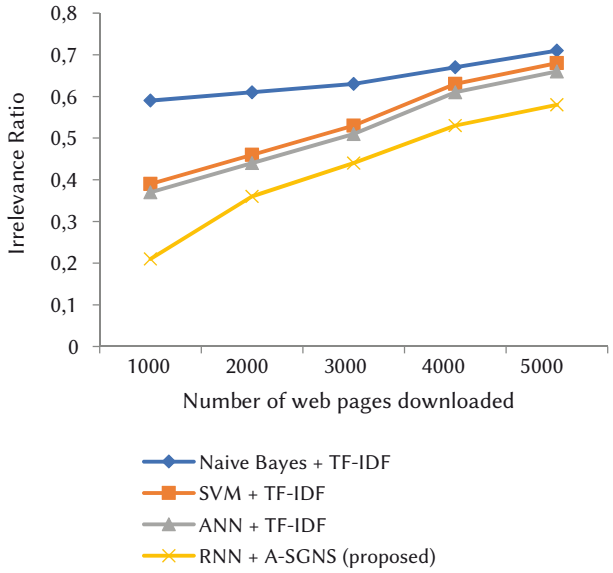

Fig. 5. Average irrelevance ratio for ten topics for the SVM + TF-IDF, NB + TF-IDF, ANN + TF-IDF and RNN + A-SGNS crawlers.

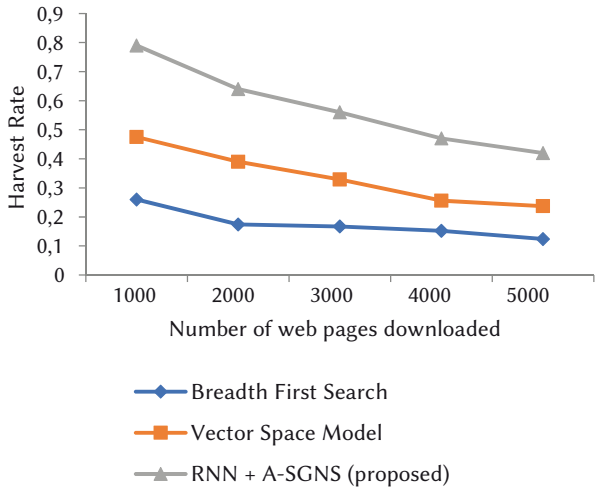

Fig. 6. Average harvest rate of ten topics for the BFS, VSM and RNN+A-SGNS crawlers.

To retrieve the associated web pages without determining their topical preferences, the breadth-first crawler explicitly selects unvisited hyperlinks. The VSM makes use of the TF-IDF to compute topical similarities but failed to capture the semantic similarity. As a result, the average harvest rate of the BFS and VSM is less than that of the RNN + SGNS and the average irrelevance ratio of the BFS and VSM is higher than that of the RNN + SGNS. Fig. 6 and Fig. 7 show the average harvest rate and average irrelevance ratio of the BFS, VSM and RNN+A-SGNS respectively. Right from the beginning, the BFS starts retrieving irrelevant results, and after 5000 web page crawls produced an average harvest rate of 0.124 and an irrelevance ratio of 0.876. The VSM crawler performed better than the BFS crawler because of the relevance computation. The VSM crawler makes use of the TF-IDF to compute topical similarities but failed to capture the semantic similarity. After 5000 web page crawls, the VSM crawler produced an average harvest rate of 0.237 and an irrelevance ratio of 0.763. The proposed RNN+A-SGNS crawler outperformed both the BFS and VSM crawlers with an average harvest rate of 0.42 and an irrelevance ratio of 0.58 after 5000 web page crawls.

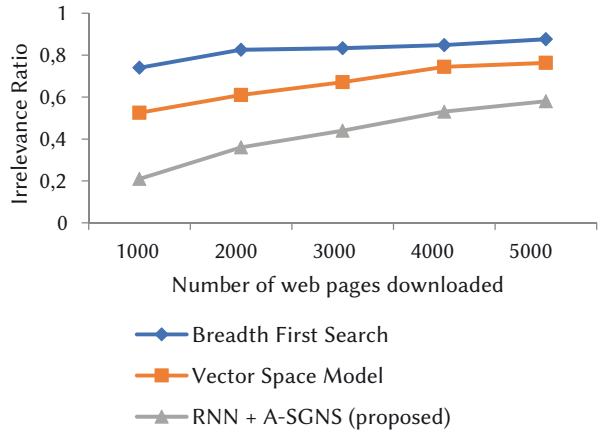

Fig. 7. Average irrelevance ratio of ten topics for the BFS, VSM and RNN+A-SGNS crawlers.

Ontology learning-based crawlers use a domain-specific ontology to ascertain the topical similarity between a topic and web pages. An ontology is a well-known representation that helps find semantic similarity. Ontologies are domain-specific and designed by domain experts. A human error in ontology design results in the retrieval of wrong results. In this work, WordNet ontology [52] for the semantic representation of words was used in the design of the optimized Naive Bayes (ONB) crawler, the ontology learning-based crawler using the ANN (OL-ANN), and the semi- supervised learning-based crawler using the SVM (SSL-SVM). Ontology learning on the dynamic internet is a difficult and time-consuming process. Given the limitations of ontologies and ontology learning, these crawlers performed poorly on the dynamic internet in terms of the harvest rate and irrelevance ratio, when compared to the proposed methodology. The ONB, ontology learning-based crawler using the ANN, the semi-supervised learning-based crawler using the SVM, and the proposed crawler produced an average harvest rate of 0.39, 0.37,0.36 and 0.42, respectively, and an average irrelevance ratio of 0.61, 0.63, 0.64 and 0.58, respectively. This clearly shows that the proposed crawler outperformed the ontology learning-based crawler. Fig. 8 and Fig. 9 show a comparison of the results of the ONB, OL-ANN, SSL-SVM and the proposed crawler in terms of the harvest rate and irrelevance ratio, respectively.

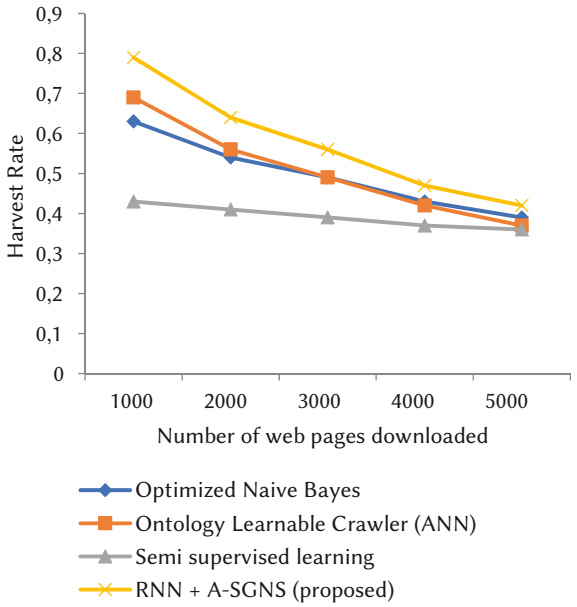

Fig. 8. Average Harvest Rate of ten topics for ONB, OL-ANN, SSL-SVM and RNN+A-SGNS.

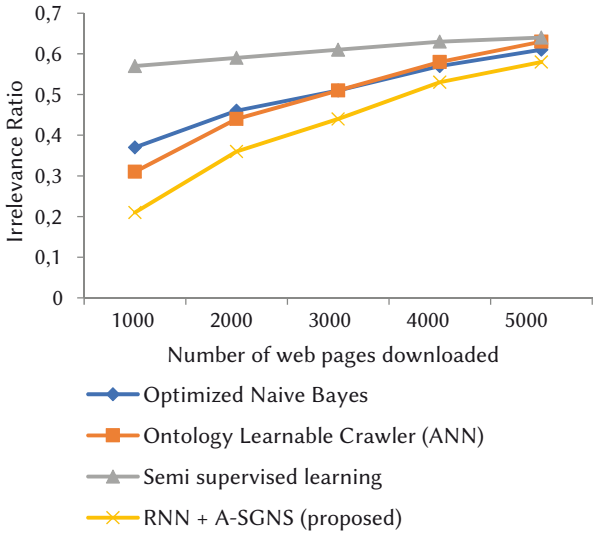

Fig. 9. Average Irrelevance Ratio of ten topics for ONB, OL-ANN, SSL-SVM and RNN+A-SGNS.

VI. Conclusion and Future Work

The A-SGNS model presented here was intended to optimize the performance of the focused web crawler. This work considers both the syntactic and semantic similarity between the topic and web page documents. The model first computes the A-SGNS model, from which the cosine similarity of the topic and document terms is calculated. The similarity vectors are given as input to the recurrent neural network to classify the web page, based on its relevance. The results of the experiment have demonstrated that the proposed system has increased the efficiency of the focused crawler, outperforming the breadth-first, VSM, and TF-IDF-based learning crawlers as well as those based on ontology learning. In conclusion, the proposed method is ideally suited to focused crawlers and has conclusively proved its efficacy.

Future directions include plans for the design of a crawler using long short-term memory networks (LSTM) or the gated recurrent unit (GRU) to resolve the long-term dependency problem of the RNN in learning sequences, brought on by problems with the vanishing gradient.

References

[1] "Internet Live Status," 2020. [Online]. Available: https://www.internetlivestats.com/total-number-of-websites/.

[2] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine BT - Computer Networks and ISDN Systems," *Comput. Networks ISDN Syst.*, vol. 30, no. 1–7, pp. 107–117, 1998.

[3] G. Ayers, J. H. Ahn, C. Kozyrakis, and P. Ranganathan, "Memory Hierarchy for Web Search," *Proc. - Int. Symp. High-Performance Comput. Archit.*, vol. 2018-Febru, pp. 643–656, 2018.

[4] Auf Wiedersehen, "The Architecture of a Large-Scale Web Search Engine, circa 2019," 2019. [Online]. Available: https://0x65.dev/blog/2019-12-14/the-architecture-of-a-large-scale-web-search-engine-circa-2019.html.

[5] B. Muller, "How search engines work: Crawling, Indexing, and Ranking," *Moz Pro*, 2020. [Online]. Available: https://moz.com/beginners-guide-to-seo/how-search-engines-operate.

[6] A. Hliaoutakis, G. Varelas, E. Voutsakis, E. G. M. Petrakis, and E. Milios, "Information Retrieval by Semantic Similarity," *Int. J. Semant. Web Inf. Syst.*, vol. 2, no. 3, pp. 55–73, 2011.

[7] Z. Liu, Y. Du, and Y. Zhao, "Focused Crawler Based on Domain Ontology and FCA," *J. Inf. Comput. Sci.*, vol. 8, no. 10, pp. 1909–1917, 2011.

[8] Z. Geng, D. Shang, Q. Zhu, Q. Wu, and Y. Han, "Research on improved focused crawler and its application in food safety public opinion analysis," *2017 Chinese Autom. Congr.*, pp. 2847–2852, 2017.

[9] T. Hassan, C. Cruz, and A. Bertaux, "Predictive and evolutive cross-referencing for web textual sources," *Proc. Comput. Conf. 2017*, vol. 2018-Janua, no. July, pp. 1114–1122, 2018.

[10] S. Chakrabarti, M. van den Berg, and B. Dom, "Focused crawling: a new approach to top-specific Web source discovery," *Comput. Networks*, vol. 31, no. 11–16, pp. 1623–1640, 1999.

[11] F. Menczer, F. Menczer, G. Pant, G. Pant, P. Srinivasan, and P. Srinivasan, "Topical Web Crawlers: Evaluating Adaptive Algorithms," *ACM Trans. Internet Technol.*, vol. V, no. February, p. 38, 2003.

[12] J. R. Park, C. Yang, Y. Tosaka, Q. Ping, and H. El Mimouni, "Developing an automatic crawling system for populating a digital repository of professional development resources: A pilot study," *J. Electron. Resour. Librariansh.*, vol. 28, no. 2, pp. 63–72, 2016.

[13] G. H. Agre and N. V. Mahajan, "Keyword focused web crawler," *2nd Int. Conf. Electron. Commun. Syst. ICECS 2015*, pp. 1089–1092, 2015.

[14] Y. Du, W. Liu, X. Lv, and G. Peng, "An improved focused crawler based on Semantic Similarity Vector Space Model," *Appl. Soft Comput. J.*, vol. 36, pp. 392–407, 2015.

[15] G. Salton, A. Wong, and C. Yang, "Information Retrieval and Language Processing: A Vector Space Model for Automatic Indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613-620, 1975.

[16] P. Bedi, A. Thukral, and H. Banati, "Focused crawling of tagged web resources using ontology," *Comput. Electr. Eng.*, vol. 39, no. 2, pp. 613–628, 2013.

[17] A. I. Saleh, A. E. Abulwafa, and M. F. Al Rahmawy, "A web page distillation strategy for efficient focused crawling based on optimized Naïve bayes (ONB) classifier," *Appl. Soft Comput. J.*, vol. 53, pp. 181–204, 2017.

[18] H. D. and F. K. Hussain, "SOF: a semi-supervised ontology-learning-based focused crawler," *Concurr. Comput. Pract. Exp.*, vol. 25, no. 6, pp. 1755–1770, 2013.

[19] H. T. Zheng, B. Y. Kang, and H. G. Kim, "An ontology-based approach to learnable focused crawling," *Inf. Sci. (Ny).*, vol. 178, no. 23, pp. 4512–4522, 2008.

[20] A. Capuano, A. M. Rinaldi, and C. Russo, "An ontology-driven multimedia focused crawler based on linked open data and deep learning techniques," *Multimed. Tools Appl.*, 2019.

[21] Y. Li, Z. A. Bandar, and D. McLean, "An approach for measuring semantic similarity between words using multiple information sources," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 4, pp. 871–882, 2003.

[22] J. Hosseinkhani, H. Taherdoost, and S. Keikhaee, "ANTON Framework Based on Semantic Focused Crawler to Support Web Crime Mining Using SVM," *Ann. Data Sci.*, 2019.

[23] D. Mukhopadhyay and S. Sinha, "Domain-Specific Crawler Design," pp. 85–112, 2019.

[24] J. Qiu, Q. Du, W. Wang, K. Yin, C. Lin, and C. Qian, "Topic Crawler for OpenStack QA Knowledge Base," *Proc. - 2017 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov. CyberC 2017*, vol. 2018-Janua, pp. 309–317, 2018.

[25] T. Suebchua, B. Manaskasemsak, A. Rungsawang, and H. Yamana, "History-enhanced focused website segment crawler," *Int. Conf. Inf. Netw.*, vol. 2018-Janua, pp. 80–85, 2018.

[26] G. Xu, P. Jiang, C. Ma, and M. Daneshmand, "A Focused Crawler Model Based on Mutation Improving Particle Swarm Optimization Algorithm," *Proc. - 2018 IEEE Int. Conf. Ind. Internet, ICII 2018*, no. Icii, pp. 173–174, 2018.

[27] H. Dong and F. K. Hussain, "Self-adaptive semantic focused crawler for mining services information discovery," *IEEE Trans. Ind. Informatics*, vol. 10, no. 2, pp. 1616–1626, 2014.

[28] W. J. Liu and Y. J. Du, "A novel focused crawler based on cell-like membrane computing optimization algorithm," *Neurocomputing*, vol. 123, pp. 266–280, 2014.

[29] P. Resnik, "Using Information Content to Evaluate Semantic Similarity in a Taxonomy," vol. 1, 1995.

[30] "Natural Language Processing Tool Kit (NLTK)," 2020. [Online]. Available: https://www.nltk.org/.

[31] E. L. and E. K. Bird, Steven, *Natural Language Processing with Python*. O'Reilly Media Inc, 2009.

[32] H. Palangi *et al.*, "Deep Sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 24, no. 4, pp. 694–707, 2016.

[33] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 4, no. January, pp. 3104–3112, 2014.

[34] T. Mikolov, M. Karafiát, L. Burget, C. Jan, and S. Khudanpur, "Recurrent neural network based language model," *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH 2010*, no. September, pp. 1045–1048, 2010.

[35] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations ofwords and phrases and their compositionality," *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2013.

[36] Y. Goldberg and O. Levy, "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method," no. 2, pp. 1–5, 2014.

[37] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *1st Int. Conf. Learn. Represent. ICLR 2013 - Work. Track Proc.*, pp. 1–12, 2013.

[38] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *COLT 2010 - 23rd Conf. Learn. Theory*, pp. 257–269, 2010.

[39] S. Ruder, "An overview of gradient descent optimization algorithms," pp. 1–14, 2016.

[40] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Inf.*, vol. 10, no. 4, pp. 1–68, 2019.

[41] G. Chen, "A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation," pp. 1–10.

[42] G. Hinton, N. Srivastava, and K. Swersky, "Lecture 6a Overview of mini-batch gradient descent," *Neural Networks Mach. Learn. Coursera*, 2012.

[43] G. van Rossum, "Python tutorial, Technical Report CS-R9526," *Cent. voor Wiskd. en Inform. (CWI), Amsterdam*, 1995.

[44] "Python 3.6," 2016. [Online]. Available: https://www.python.org/downloads/release/python-360/.

[45] "Spyder IDE," 2009. [Online]. Available: https://www.spyder-ide.org/.

[46] L. Richardson, "Beautiful Soup Documentation Release 4.4.0." 2019.

[47] "urllib," *Python*, 2020. [Online]. Available: https://docs.python.org/3/library/urllib.html.

[48] "lxml parser," *Python*, 2020. [Online]. Available: https://lxml.de/elementsoup.html.

[49] G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Mueller, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 1, pp. 2825–2830, 2011.

[50] D. Isa, L. H. Lee, V. P. Kallimani, and R. Rajkumar, "Text document preprocessing with the bayes formula for classification using the support vector machine," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 9, pp. 1264–1272, 2008.

[51] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent Advances in Recurrent Neural Networks," pp. 1–21, 2017.

[52] Princeton University, *"About WordNet." WordNet.* Princeton University, 2010.

P. R. Joe Dhanith

P.R.Joe Dhanith received his B.Tech degree in Information Technology from Anna University in 2010 and M.E degree in Computer Science and Engineering from Anna University in 2012. He is currently pursuing his Ph.D degree in Computer Science and Engineering at National Institute of Technology Puducherry. His main research interests includes web mining, web crawling and information retrieval.

B.Surendiran

B. Surendiran is currently working as an Assistant Professor in the Department of Computer Science and Engineering at National Institute of Technology Puducherry, Karaikal, India. He completed his Ph.D in Computer Science and Engineering at National Institute of Technology Tiruchirapalli. His research interest includes recommender systems, machine learning and data mining. He has published more than 30 papers in international journals.

S.P.Raja

S. P. Raja completed his B. Tech in Information Technology in the year 2007 from Dr. Sivanthi Aditanar College of Engineering, Tiruchendur. He completed his M.E. in Computer Science and Engineering in the year 2010 from Manonmaniam Sundaranar University, Tirunelveli. He completed his Ph.D. in the year 2016 in the area of Image processing from Manonmaniam Sundaranar University, Tirunelveli. His area of interest is image processing and cryptography. He is having more than 13 years of teaching experience in engineering colleges. Currently he is working as an Associate Professor in the department of Computer Science and Engineering in Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai. He published 30 papers in International Journals, 24 in International conferences and 12 in national conferences. He is an Editorial Board Member of International Journal of Interactive Multimedia and Artificial Intelligence.