IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received April 27, 2021, accepted May 23, 2021, date of publication June 4, 2021, date of current version June 21, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3086586

High-Quality Train Data Generation for Deep Learning-Based Web Page Classification Models

JEONG-JAE KIM[1], BYUNG-WON ON[2], AND INGYU LEE[3]
[1]Graduate School of Cognitive Science, Yonsei University, Seoul 03722, South Korea
[2]Department of Software Convergence Engineering, Kunsan National University, Gunsan 54150, South Korea
[3]Sorrell College of Business, Troy University, AL 36082, USA

Corresponding author: Byung-Won On (bwon@kunsan.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant by Korean Government through the Ministry of Science and ICT (MSIT) under Grant NRF-2019R1F1A1060752.

ABSTRACT The current deep learning models detecting relevant web pages show low accuracy because of the poor quality of the training data. In this paper, we propose a novel algorithm to automatically generate high-quality training data based on the frequency of the document including the entity of interest. Our experimental results with movies and cellphones data sets show that the average F_1-score of the deep learning models (FNN, CNN, Bi-LSTM, and SeqGAN) trained with our proposed algorithm shows up to 0.9992 in F_1-score.

INDEX TERMS Text classification, deep learning, automatic labelling.

I. INTRODUCTION

With the advent of the fourth industrial revolution, Artificial Intelligence (AI)–based data mining algorithms play a key role in extracting unknown but informative knowledge from big data to improve enterprise productivity and bring out technological innovation. To apply the AI-based data mining algorithms, a lot of relevant data on the Internet should be automatically collected, categorized, and labelled. The results are stored as in a knowledge base of the entity without human intervention, and used in the data mining algorithms. The classification algorithm to detect the relevance web pages plays an important role in the latter process. However, all accuracy values of the existing classification models are poor [1] because all discriminative features should be directly exploited by domain experts who may leave out key features by mistake. Additionally, it is known that such methods do not work well to address the non–linear classification problem according to data complexity.

To improve the accuracy of the models, various AI–based models (e.g., Feed–forward Neural Network (FNN), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN)) have been considered in recent time. In the AI–based models, there are two different approaches to avoid the overfitting problem that degrades the accuracy of the deep learning models including FNN, CNN, and RNN [2]. The first approach is to reduce the complexity of the deep learning models. The other is to use high–quality training data. In the former approach, deep learning models themselves have high complexity with a number of hidden units, weights, and bias parameters. The accuracy would be decreased if we reduced the complexity of the models. On the other hand, our work focuses mainly on the latter approach. As a good example, the accuracy of Google translation service has recently improved significantly because a sequence–to–sequence model, one of RNN–based deep learning models, is well trained with a huge number of high–quality training data including one billion pairs of Chinese sentences that match English sentences. Back to our problem, it is non–trivial to obtain high–quality training data in which each web page has its class label indicating that it is relevant or not with a target entity. Even though we collect many web pages automatically, human evaluators should manually check each web page to determine whether it is actually relevant or not. In this way, collecting high–quality training data by human judgement is fairly limited.

In this article, given a target entity and a set of web pages, we propose a novel automatic algorithm for **High-quality Training data Generation (HiTGen)**, thereby considerably improving the accuracy of the existing deep learning models. We also show that the pseudo–generated training data almost match the training data made by human. Throughout this article, our proposed algorithm depicted in Figure 1 is called HiTGen which is working based on the principle that web

The associate editor coordinating the review of this manuscript and approving it for publication was Qichun Zhang.

pages with high frequency are relatively relevant with a target entity rather than ones with low frequency.

In the proposed method, given a pre-defined entity $e = \{a_1, a_2, \ldots, a_n\}$, where a_i is one of the attributes, we create all possible queries using the attribute values of e. For each query, top-k web pages are retrieved through a certain search engine. If a web page w is retrieved by many queries, we consider it to have high frequency. According to our hypothesis, if w has high frequency, it is considered to be relevant with e. This is, the class label of w is relevant. We also extract top-l words that have high TFIDF values and frequently appear in web pages with high frequency and then assign them to the feature set. With the class label and the feature set, we automatically make the high-quality training set for deep learning based web page classification models. We will discuss the details in Section 3.

The contributions of our work are as follows:

- To boost the existing deep learning models, we propose a novel algorithm of **automatically generating high–quality training data called HiTGen**. The accuracy of the existing deep learning models such as FNN, CNN, and RNN trained with HiTGen can be largely improved without reducing the complexity of the deep learning models. To the best of our knowledge, this is the first study to automatically collect high–quality training data based on the frequency information of web pages for the deep learning models in the web page classification problem.
- To evaluate the proposed method, we experimented 13 methods – (i) conventional classification models such as Support Vector Machines (SVM) [3], Random Forest (RF) [4], and AdaBoost (AB) [5]; (ii) existing deep learning models such as FNN [6], CNN [7], Bidirectional Long Short-Term Memory (Bi–LSTM) [8], and Sequential Generative Adversarial Networks (SeqGAN) [9]; and (iii) all models using HiTGen. Our experimental results show that all models using HiTGen outperform all existing learning models.
- We also experimented two different data sets – movies and cellphones. Our experimental results show that the deep learning models using HiTGen outperform other existing learning models across different data sets. This implies that any deep learning model using HiTGen works well across the domains in which each entity is represented by its attributes.

The remainder of this article is organized as follows: First, we introduce existing methods related to this work. In particular, we discuss the novelty of our method and the main difference between previous studies and our work. In Section III, we formally define the research problem. Then, we describe the details of the proposed algorithm to boost the existing deep learning models in Section IV. We explain the experimental set–up in Section V and then discuss the experimental results in Section VI. We also discuss the findings and their implications in Section VII. Finally, we summarize our work followed by the future research direction in Section VIII.

II. RELATED WORK

Our research is related to information retrieval [10], [11], pseudo-relevance feedback [12], [13], diversified search [14], knowledge graph [15], and entity search [16]. We focus more on web page classification using a deep learning model in this paper. Therefore, we introduce the existing studies of web page classification in section II-A and the research trends of text classification using deep learning in section II-B.

A. WEB PAGE CLASSIFICATION

Focusing on the subject of a web page, it can be divided into a classification problem of which category the web page belongs to and a detection problem of which event/action the web page belongs to. In the classification problem, genre classification [17], controversy classification [18], and emotion classification [19] have been proposed. Rumor detection [20], phishing detection [21], fraud detection [22], and fake detection [23] have been proposed in the detection problem. Our research is a classification problem as to whether the web page is related to the entity of interest. In the web page classification, labeling a given web page by summarizing its content is an important research issue [24]. presented Multi-Label Random Forest (MLRF) that can quickly generate several labels such as 'dominos', 'dominos pizze', and 'domino pizza online' when a web page relevant with 'dominos pizza' is given as input. To improve the accuracy of web page classification, [25] proposed an approach that incorporates web site-dependent priors appearing in web pages centered around a certain topic like sports and the topical structure of the web (e.g., Hyperlink graph). [26] presented Learning Quality Soft Clustering (LQSC) and Learning Quality Hard Clustering (LQHC) that extract quality and quantity features from training data. Text length, illumination, and video quality are some of such features. Then, SVM classifier was trained with the features for web page classification. The authors reported that their proposed method is slightly better than SVM classifier without the features. Among the above methods to classify a collection of web pages, [26] is a little close to our method. However, unlike [26], our proposed method does not need the content-based features because the deep learning model is working without features selected by the help of domain experts.

B. DEEP LEARNING FOR TEXT CLASSIFICATION

Many methods such as graphical model [27], hierarchical structure [28], feature engineering [29], [30] have been proposed for text classification. In terms of model architecture, CNN [31] and LSTM [32] have been used. Recent studies have trained huge corpus on language models with extremely high complexity, increasing the performance of the Natural Language Processing (NLP) field through transfer learning. Many applications require large amounts of labeled data for fine-tuning, but this is challenging in terms of time and cost. Extreme Multi-label Text Classification (XMTC), which tags specific text with multiple highly relevant labels from labeled bulk data, has emerged as an important issue. To address this, [33] captures the most relevant text portion of each label with an attention mechanism in raw text with richer semantic context information and utilizes a shallow and wide Probabilistic Label Tree (PLT) to handle millions of labels. [34] proposes self-training based on uncertainty estimates of neural networks using large unlabeled data. [34] is semi-supervised learning, assumptions about the initial model must be sufficient, and high-quality labeled data is required. On the other hand, our method is unsupervised learning, which automatically generates training data for both relevance and irrelevance to the entity of interest from unlabeled data.

III. PROBLEM STATEMENT

In this section, before we define our train set auto-generation problem, we first define an *entity* and *deep learning based web page classification problem* as follows:

> **Definition of entities**: An entity e_i is a real-world object and its examples are a person, a citation, a MP3 file, a company, etc. The entity has two main properties – (1) Identifier (ID) and (2) Content. ID is the description of the entity (mainly the entity name) and Content is a set of attributes describing e_i. This is, $e_i = \{a_1, a_2, \ldots, a_n\}$. As an example of entities, e_i=(ID:“John Smith”, Content:{“Stanford U.”, “650-721-1444”, “Data Mining”}), where the three attributes stand for workplace (a_1), phone number (a_2), and major (a_3).

Based on the entity definition above, we formally define the web page classification problem as follows:

> **Definition of web page classification**: Given an entity (e_i) of interest, where $e_i \in$ Domain $D = \{e_1, e_2, \ldots, e_n\}$, collect top-$k$ important web pages ($W = \{w_1, w_2, \ldots, w_k\}$) that include e_i. In this work, how the top-k web pages are determined is out of scope but one of plausible algorithms is a list of web pages order of Google's PageRank. For $w_i \in W$, automatically determine whether w_i is *really* relevant with e_i or not.[1] In this work, we focus merely on deep learning-based models (i.e., SVM, RB, and AdaBoost) that outperform traditional classification models (FNN, CNN, and Bi-LSTM) as shown in Table 5.

This problem is significantly challenging because of several reasons. Firstly, many entities are ambiguous. An entity like ‘troy’ may indicate one of a history event, a city, a movie, and a university. Secondly, the number of web pages containing a target entity is at most hundreds. Even such web pages are relevant to several other entities (a history event, a city, and a university) rather than the entity (a movie). In other words, the web pages are grouped to the four clusters. One cluster is a set of web pages relevant with a history event, Another is a cluster of web pages relevant with a city, and so on. In addition, the numbers of the web pages belonging to those entities are unbalanced. These points are likely to significantly degrade the quality of training data to address the deep learning based web page classification problem. As a result, the deep learning models are likely to cause the overfitting problem or to show low accuracy.

Finally, in order to improve the accuracy of the deep learning models, we clearly define the following problem.

> **Definition of automatic train set auto-generation**: Given a pre-defined entity $e_i = \{a_1, \ldots, a_n\}$, automatically generate a high-quality training set for existing deep learning models, where the training set contains a few feature vectors. A feature vector corresponds to a web page w. In each feature vector v, $v[0]$ is class label (relevance or irrelevance) and $v[i]$ is a weight (importance) of the most discriminative word in w.

IV. MAIN PROPOSAL

To address the problem, we propose a novel *training data pseudo–generation* algorithm, using which the accuracy of deep learning–based web page classification models is improved largely. It is no doubt that any other deep learning model shows high accuracy if it is learned with high–quality training data. In the following subsections, we will discuss the proposed method in detail.

A. A HIGH–QUALITY TRAINING DATA GENERATION METHOD FOR DEEP LEARNING MODELS

In the problem, given an entity $e = \{a_1, a_2, \ldots, a_m\}$, where a_i stands for the i–th attribute that describes e, top-k web pages are retrieved by one of general–purpose search engines such as Google. In the top–k web pages, some pages will be relevant with e but other pages will not. In even some cases, it is difficult to determine if a web page is really relevant with e. In this set–up, the real problem is how machine on behalf of human can correctly determine the relevance of a number of web pages retrieved by the search engine because our final goal of this research is to construct a high–quality training data.

Recent outstanding achievements based on deep learning in the fields of image processing and machine translation are based on (1) high–performance computer resource such as GPGPU and (2) big data accumulation for training deep learning–based statistical models (DLM). The more high–quality training data is, the better the accuracy of DLM is. However, unfortunately, it is non–trivial to collect a high–quality training data in real–world. The training data is a set of pairs, each of which consists of (class, vector). For example, in our context, the class is either relevant or

[1] It is possible that a web page is relevant to multiple entities. For example, Harry Potter may represent both movie and novel. In this case, we assume that the movie (e_1) and novel (e_2) of Harry Potter are different entities. Most web pages are likely to be relevant with either of them. If a web page w mainly explains an entity e_1 more than the other entity e_2, we will consider that w is relevant with e_1.

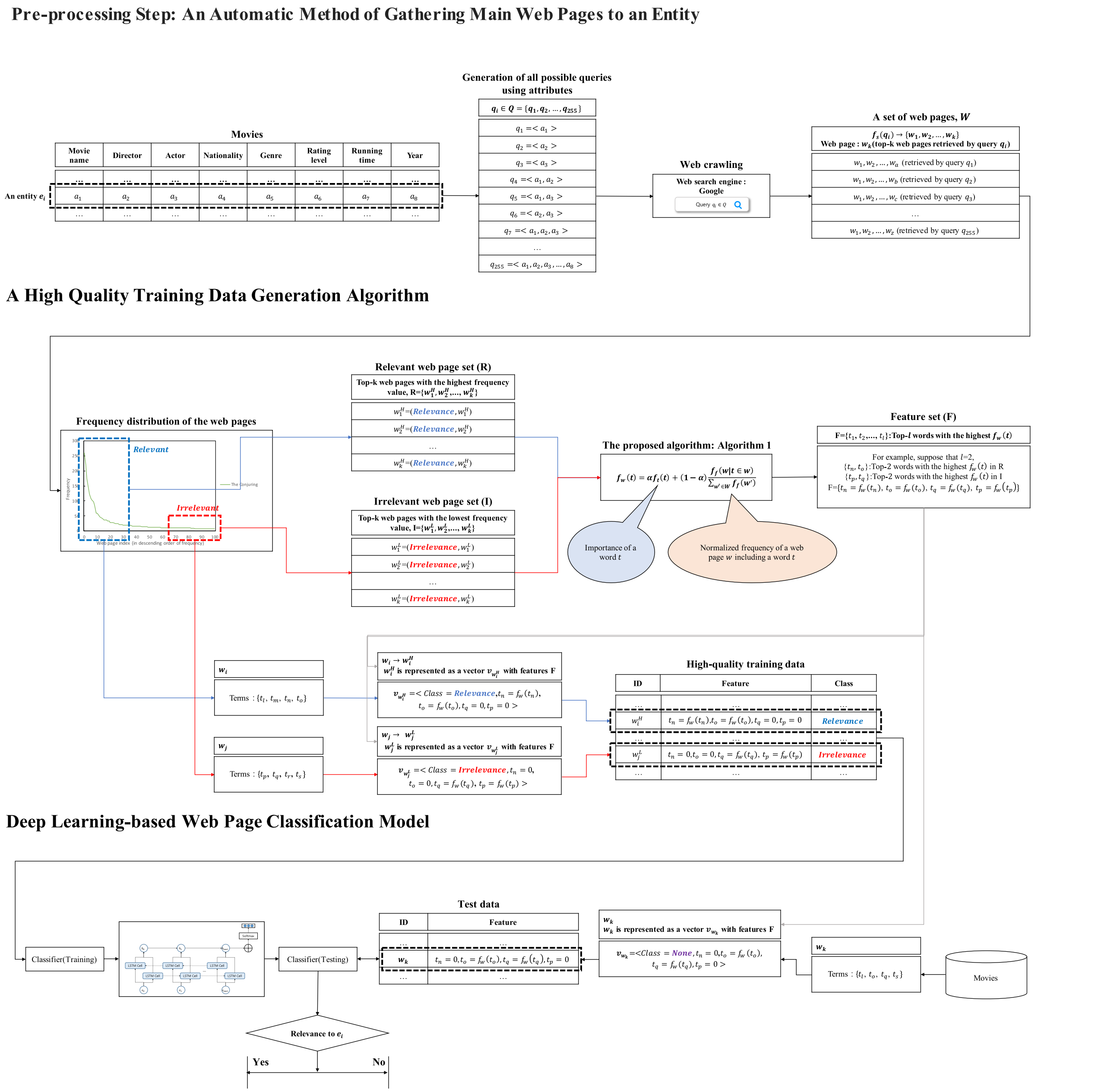


FIGURE 1. Overview of our proposed method.

irrelevant with the target entity, and the vector is composed of l word features, where l is the number of vocabularies (unique words) in a collection of web pages. Given a web page as input, it is transformed to a vector. So far, in existing models, the class of the vector is manually determined as either relevant or irrelevant. If the i–th $(0 \leq i \leq l)$ vocabulary appears in the web page, the i–th entry is 1 and 0, otherwise. This vector is often called one–hot vector. In many cases, the entry is filled with the frequency or weight (importance) of the vocabulary. To determine the class per pair, domain experts should label vectors manually so this task is labor–intensive and time–consuming, thereby collecting a large number of training data is extremely limited in industrial sites. As a result, the lack of high–quality training data that are pairs of (class, vector) will reduce the accuracy of DLM in real applications.

To *automatically* generate high-quality training data, we propose a novel pseudo–labelling algorithm for deep learning–based models. Table 1 shows the notation terms for explaining our proposed algorithm. For instance, an entity e and its attributes $\{a_1, a_2, a_3\}$ are considered as e (ID)=''John Smith'', a_1 (Workplace)=''Stanford University'', a_2 (Phone number)=''650-721-1444'', and a_3 (Major)=''Data Mining''. Given this entity as input, we create all possible search

TABLE 1. Notation terms for describing HiTGen.

Term	Description
e	An entity of interest
a_i	The i–th attribute of e
q_i	The i–th search query
w_i	The i–th web page
t	A term in a web page
$f_q(e)$	This function returns all possible query combinations using the attributes of e
$f_s(q_i)$	This function returns top–k web pages retrieved by q_i
$f_f(w_i)$	This function returns the frequency value of w_i
$f_t(t)$	This function returns the weight (importance) of t by TF/IDF
$f_w(t)$	This function returns the weight (importance) of t by Eq. (1)

queries by function $f_q()$ that generates combinations of e's attributes. For example, $f_q(e = \{a_1, a_2, a_3\}) \rightarrow \{q_1 =< a_1 >, q_2 =< a_2 >, q_3 =< a_3 >, q_4 =< a_1, a_2 >, q_5 =< a_1, a_3 >, q_6 =< a_2, a_3 >, q_7 =< a_1, a_2, a_3 >\}$. For each query $q_i \in Q = \{q_1, \ldots, q_7\}$, through a search engine (e.g., Google) $f_s()$, top–k web pages are retrieved. This is, $f_s(q_i) \rightarrow \{w_1, w_2, \ldots, w_k\}$, where w_1 is ranked higher than w_2 and ranked much higher than w_k in the output of the search engine.

Interestingly, through our intensive experiments, we observed that **web pages with high frequency are usually more relevant than ones with low frequency**. If a web page w_1 is retrieved by both q_1 and q_5, then the frequency of w_1 is 2 (i.e., $f_f(w_1) = 2$). Based on this pattern, we define the correlation between frequency of a given web page and relevance with e as the following hypothesis.

- $f_f(w_i) = m$ implies that the frequency value of the web page w_i is m.
- If $f_f(w_i) > f_f(w_j)$, then the web page w_i is more relevant with e than the web page w_j.

Our proposed pseudo–labelling algorithm is based on the above hypothesis. The reason why web pages with high frequency are highly relevant with e compared with ones with low frequency is: If each query describes an entity e in part, then a relatively large number of queries will explain e better. Technically, assuming that each query corresponds to an axis in the multi–dimensional coordinate system, let us back to the above example of $e = \{a_1, a_2, a_3\}$. If a web page w_1 is retrieved by all queries, then such axes indicate that the extent of w relevant with e is $\frac{7}{7} = 1$, while that of another web page w_2 is $\frac{2}{7} = 0.29$ if w_2 is retrieved by only two queries q_1 and q_4.

Figure 2(a) and (b) show the frequency value of the web pages per entity. In the figure, the x-axis means web page identifiers of an entity in the descending order, while the y-axis means the web pages' frequencies. The figures clearly show that the top-10 web pages are likely to have the highest frequency values. The figures show power law distribution in which a few web pages in the left side have high frequency, while most web pages in the right side have low frequency. According to the results of the manual investigation, it turns out that almost all web pages are irrelevant with the target entity except a few web pages with high frequency. To show that our hypothesis is statistically significant, we will discuss in detail in Section IV-C.

As W is denoted by the set of web pages retrieved by $f_s(q_1), \ldots, f_s(q_{(2^{|\#\text{ of attributes}|}-1)})$, $|W| \leq k(2^n - 1)$, where $|W|$ is the number of web pages in W. The weight (importance) value of a word t in a web page $w_i \in W$ is computed by function $f_t(t) = \frac{f_{t,w_i}}{\Sigma_{t' \in w_i} f_{t',w_i}} log \frac{|W|}{|\{w \in W | t \in w\}|}$, where f_{t,w_i} is the number of occurrences of t in w_i. In other words, $f_t(t)$ computes the Term Frequency / Inverse Document Frequency (TF/IDF) of t.

To generate the pair (class, vector) to a web page $w \in W$, which is an individual data in the training set that is necessary to learn deep learning models, we need to select l main vocabularies from W that are used as the feature of vectors. In our approach, we quantitatively compute the weight values of all words in W and then select top–l main words with the highest weight values. To estimate the weight value of each word based on our proposed hypothesis, we propose Eq. (1) in which we formally define a new equation $f_w()$ of computing the weight value of a word t in addition to $f_t()$.

$$f_w(t) = \alpha f_t(t) + (1 - \alpha) \frac{f_f(w | t \in w)}{\sum_{w' \in W} f_f(w')} \quad (1)$$

Using $f_w(t)$, we can quantify both how important t is in W and how relevant t is with e through $f_f(w)$ subject to $t \in w$. Especially, in the above equation, the first term is measuring the TF/IDF of t. In the second term, if t appears in w with high frequency, t is weighted more than any other word in web pages with low frequency. The ratio (importance) of the first and second terms can be changed by using α value ($\alpha = 0.7$ in our experiment). After $f_w(\forall t \in W)$s are computed, top–l words with the highest $f_w()$ values are chosen as features, using which vectors to web pages $\in W$ are generated.

Now we have top–l word features and call them "feature set (F)" here. The words in the feature set are rearranged in the descending order by $f_w(t)$. In the next step, top–k web pages with the highest frequency values are selected and these web pages are automatically marked as relevant web pages – $\{w_1^H, w_2^H, \ldots, w_k^H\}$. For example, "Relevant web page set (R)"= $\{$(Relevance, w_1^H), (Relevance, w_2^H), …, (Relevance, w_k^H)$\}$. Similarly, top–k web pages with the lowest frequency values are chosen and these web pages are automatically marked as irrelevant web pages – $\{w_1^L, w_2^L, \ldots, w_k^L\}$. For example, "Irrelevant web page set (I)"= $\{$(Irrelevance, w_1^L), (Irrelevance, w_2^L), …, (Irrelevance, w_k^L)$\}$.

For a web page $w_i^H \in R$, w_i^H is represented as a vector $v_{w_i^H}$ with features F. For example, suppose that $F = \{t_1, t_3, t_l\}$ and $w_i^H = \{t_1, t_2, t_3, t_4\}$, where t_1, t_2, t_3, and t_4 are the words in w_i^H, $v_{w_i}^H =<$ Class = Relevance, $t_1 = f_w(t_1), t_3 = f_w(t_3), t_l = 0 >$. In the same way, for a web page $w_i^L \in I$, w_i^L is represented as a vector $v_{w_i^L}$ with features F. For example, suppose that $F = \{t_1, t_3, t_l\}$ and $w_i^L = \{t_5, t_7, t_l, t_{100}\}$, where t_5, t_7, t_l, and t_{100} are the words in w_i^L, $v_{w_i}^L =<$ Class = Irrelevance, $t_1 = 0, t_3 = 0, t_l = f_w(t_l) >$.

Deep learning–based models are trained with these vectors like $v_{w_i}^H$ and $v_{w_i}^L$.

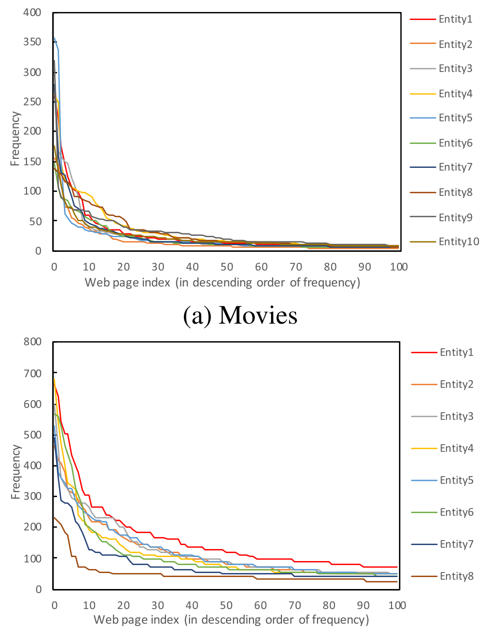
Algorithm 1 HiTGen: Automatic High–quality Training Data Generation for Deep Learning Models

$Q = f_q(e = \{a_1, \ldots, a_n\})$;
$W = \phi$;
for $q \in Q$ **do**
  $W = W \cup f_s(q)$;
for $w \in W$ **do**
  // m: Frequency value of a web page w
  $m = f_f(w)$;
R = top–k web pages with the highest m;
for $w \in R$ **do**
  for $t \in w$ **do**
    // s_R: Weight value of a word t in R
    $s_R = f_w(t)$ in Eq. (1) ;
I = top–k web pages with the lowest m;
for $w \in I$ **do**
  for $t \in w$ **do**
    // s_I: Weight value of a word t in I
    $s_I = f_w(t)$ in Eq. (1) ;
$F_R = \{t_{\text{top-1}}, t_{\text{top-2}}, \ldots, t_{\text{top-}l}\}$: top–$l$ words with the highest s_R;
$F_I = \{t_{\text{top-}l}, t_{\text{top-}(l\text{-}1)} \ldots, t_{\text{top-1}}\}$: top–$l$ words with the highest s_I;
$F = F_R \cup F_I$;
for $w \in R$ **do**
  $w \rightarrow$ vector $v_w^H =<$ Class=Relevance $>$;
  for $t \in w$ **do**
    if $t \in F$ **then**
      $v_w^H = v_w^H \cup < F(t) = f_w(t) >$;
  Add v_w^H to the training set for deep learning models;
for $w \in I$ **do**
  $w \rightarrow$ vector $v_w^L =<$ Class=Irrelevance $>$;
  for $t \in w$ **do**
    if $t \in F$ **then**
      $v_w^L = v_w^L \cup < F(t) = f_w(t) >$;
  Add v_w^L to the training set for deep learning models;

The advantage of the proposed method in Algorithm 1 is that we can automatically generate high–quality training data for deep learning models. The usage of these training sets will greatly improve the accuracy of existing deep learning models. In this work, to address the web classification problem, together with training data generated by Algorithm 1, we use various deep learning models including FNN, CNN, Bi–LSTM (as the best among RNN models), and SeqGAN to see how much such models are improved.

B. TIME COMPLEXITY OF ALGORITHM

Algorithm 1 shows the pseudo code of our proposed method called HiTGen. In line 1, $2^n - 1$ queries to an entity e are created by the function $f_q()$, where n is the number of attributes per entity. In our context, because n is less than 10, the time complexity is considered as $O(1)$. In line 2, the set W is initialized so that it takes $O(1)$. In line 3 $\sim$ 4, the time complexity is $O(|Q|)$, where $|Q|$ stands for the number of queries ($|Q| = 2^n - 1$), because top-10 web pages are retrieved by a query q and are added to W. In line 5 $\sim$ 13, the time complexity is $O(|W|)$, where $|W|$ means the number of web pages retrieved by all queries, because the frequency values of all web pages in W are computed. In line 14 $\sim$ 17, the weight value of each term in all web pages in W is computed by Eq. 1. Thus, the time complexity is $O(|T||W|)$, where $|T|$ indicates the number of terms per web page. In line 18 $\sim$ 20, because the sets F, R, and I are initialized, the time complexity is $O(3)$. In line 21 $\sim$ 26, if each term w in the web pages with the highest frequency values belongs to the set F, add w to a vector as a feature. Therefore, the time complexity is $O(|K||T|)$, where $|K|$ means the number of web pages in the set R. The pseudo code between line 27 $\sim$ 32 is the same as that between line 21 $\sim$ 26. As a result, the final time complexity of Algorithm 1 is $O(2|K||T| + |T||W| + |W| + |Q| + O(5))$. Even though the input size $\rightarrow \infty$, $|Q|$, $|T|$, and $|K|$ are not almost changed and can be considered to be constant values, whereas $|W|$ approaches ∞. Thus, we can summarize the time complexity of Algorithm 1 as $O(|W|) = O(N)$.



(a) Movies

(b) Cellphones

FIGURE 2. The frequency of ten entities chosen at random in two data sets.

C. STATISTICAL VERIFICATION OF OUR HYPOTHESIS

To test if our hypothesis is statistically significant, we first assume two propositions:

- *Proposition P_1: A web page w has high frequency*; and
- *Proposition P_2: w is relevant with a target entity*.

To investigate that P_1 is strongly correlated with P_2, we selected ten entities chosen at random in the movie data

TABLE 2. Result of t–test.

Test	G_1			G_2			p-value of Levene's test	t-Score	p-value
	N	Mean	Std	N	Mean	Std			
Test I	30	10	0	30	9.975	0.006	0.078	$1.795 < \alpha\_58, 0.025 = 2.002$	0.078
Test II	30	10	0	30	9.967	0.007	0.04	$2.112 > \alpha\_58, 0.025 = 2.002$	0.043
Test III	8	10	0	8	9.781	0.115	0.06	$1.825 < \alpha\_14, 0.025 = 2.145$	0.089
Test IV	8	10	0	8	9.813	0.085	0.1	$1.821 < \alpha\_14, 0.025 = 2.145$	0.09

TABLE 3. Data characteristics.

Data set	Movies	Cellphones
# of entities	30	8
# of web pages	16,397	1,308
# of clusters in all web pages	187	64
Avg. # of web pages per cluster	88	20

TABLE 4. Experimental set–up of the used models (SVM: Optimal trade–off value between training error and margin, RF : # of trees and max depth of the tree, AB : # of trees, max depth of the tree, and learning rate).

Methods	Experimental set–up
SVM	0.001∼10
RF	10, None
AB	10, None, 1
FNN	Batch size=5∼50, Adam optimizer(learning rate=0.001), dropout rate=0.98, 5 hidden layers H_1, H_2, H_3, H_4, and H_5 (H_1 contains 1,000 units; H_2 contains 800 units; H_3 contains 600 units; H_4 contains 400 units; and H_5 contains 200 units)
CNN	Batch size=5∼100, Adam optimizer(learning rate=0.001∼0.1), dropout rate=0.5∼1.0, 1D–Convolutional layer(window size=4∼128, stride=2), 1D-Max-pooling layer(kernel size=2, stride=2), # of feature maps=4∼8
Bi–LSTM	Batch size=5∼100, Adam optimizer(learning rate=0.0001), dropout rate=1.0, # of units in the LSTM cell=128

set. For each entity, we collected top–10 relevant web pages (T) and top–10 irrelevant web pages (B) retrieved by $f_f(w_i)$. Then, four human evaluators manually labelled with the same criteria whether each web page is relevant or not. The criterion for determining relevance is whether the main subject of each web page is the target entity, even though the main content is part of the web page [35]. As a result of the inter–rater reliability test, the average of the Fleiss' Kappa values was 0.96, and the evaluators' labelling is observed as ''almost perfect agreement''.

Preparing a variable $R = 0$, each human evaluator added 1 to R if he/she decided that each web page in T is relevant, and added 0 to R, otherwise. Now, we used hypothesis testing on the mean values of two groups. Group 1 (G_1) indicates that all top–10 web pages per entity in the population are always relevant (i.e., $\frac{n\times 10}{n}$), where n stands for # of entities in the population. On the other hand, Group 2 (G_2) indicates that # of actually relevant web pages in T per entity in the population (i.e., $\frac{\Sigma_{i=1}^{n} R_i}{n}$). Since the variance of the population is unknown, we used t–test with the two groups. Test I is the t–test of T in the movies data set. When we first tested Levene's test using IBM–SPSS Statistics 21, it resulted in $\sigma_1^2 = \sigma_2^2$ because of p–value $= 0.078 > \alpha = 0.05$.

Test I: t–test under homoscedasticity of variance. The means of Group 1 and Group 2 are μ_1 and μ_2, respectively.

- $H_0 : \mu_1 - \mu_2 = 0, H_1 : \mu_1 - \mu_2 \neq 0$ (significance level $\alpha = 0.05$)
- Sample means $\bar{G_1} = 10, \bar{G_2} = 9.975$
- Sample deviations $S_1^2 = 0, S_2^2 = 0.006, S_p^2 = 0.003$
- t-Score $T = \frac{\bar{G_1}-\bar{G_2}}{S_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} = \frac{10-9.975}{0.054\sqrt{\frac{1}{30}+\frac{1}{30}}} = 1.795 < \alpha_{58,0.025} = 2.002$ (p-value $= 0.078 > \alpha = 0.05$)
- Therefore, H_0 is not rejected because the mean value of G_1 is statistically equal from that of G_2.

The above t–test result shows that Group 1 is not statistically different from Group 2. In addition, using power.t.test() function in R programming language, we computed the power value of the t–test. Our experimental result shows that the power value is 1, indicating that the t–test is free from Type2 error.

In the table, Test II is the t–test of B in the movie data set, and Test III and Test III are the t–tests of T and B in the cellphone data set. All three of these experiments showed similar results in Test I. Because of space limitation, we leave out the details of the other tests.

V. EXPERIMENTAL SET–UP

In the previous section, we described the details of the proposed algorithm for getting high–quality training data required in various deep learning models. From now on, we will introduce the process of evaluating main deep learning models based on the proposed method, comparing to both conventional classification models and deep learning models as the baseline method with two different data sets – (1) Movies [36] and (2) Cellphones [35]. Such data sets are well–known as benchmark data. Table 3 shows the brief characteristics of the data sets in which all words were replaced by lower–case letters after images, moving pictures, and advertising texts were filtered. Then, stop words[2] in the all web pages were removed and derived words were converted to root forms through a stemming software.[3] To select the discriminative features of input vectors of conventional classification methods, we first computed TF/IDF values of all words in each data set, and then used top–k words with the highest TF/IDF value as the feature set. Through intensive experiments, we decided that the number of the words in the feature set in the movie data set was 40 as the optimal number. Similarly, the number of the words in the feature set in the cellphone data set was 40. Finally, after making input vectors based on the feature set in each data set, we converted the input vectors to the word embedding vectors, which is the input of the models used in our experiments, using Word2Vec.[4]

[2] https://en.wikipedia.org/wiki/Stop_words
[3] https://tartarus.org/martin/PorterStemmer/index.html
[4] https://deeplearning4j.org/docs/latest/deeplearning4j-nlp-word2vec

We implemented the proposed algorithm in addition to FNN, CNN, Bi–LSTM, and SeqGAN models in Python and TensorFlow.[5] The experimental set–up of all methods used in our experiment is summarized in Table 4. Through our intensive experiments, we found the optimal values of the hyper parameters that are suitable for our problem. For the initial values of weight parameters, we used the truncated normal method [37]. As an activation function, ReLU was used in the entire layers except the output layer in which the activation function was SoftMax function. We also made use of cross entropy as loss function. To improve the accuracy of the models, we used dropout and regularization techniques in addition to Adam optimizer for carrying out backward propagation of errors. After completing the implementation of the deep learning models, we attempted to find the best dropout and learning rates.

To validate the effectiveness of the proposed algorithm, we compared the learning models boosted by HiTGen to the existing FNN, CNN, Bi–LSTM, SeqGAN, SVM,[6] RF and AB.[7] Through four cross–validation in the training step, all web pages collected for each entity were divided into four run sets. Each model had been first trained with the three run sets and then classified each web page in the rest set to either relevant or irrelevant class. Changing the order of the run sets, we performed the train and test steps four times, and measured the average precision, recall, and F_1–score of each model.

All models were in standalone executed in a high-performance workstation server with Intel Xeon 3.6GHz CPU with eight cores, 24GB RAM, 2TB HDD, and TITAN-X GPU with 3,072 CUDA cores, 12GB RAM, and 7Gbps memory clock.

For the evaluation metric, we used precision, recall, F_1–score measures that have been widely used in IR community. The reason is that measuring the accuracy of each model is not informative if class distribution is imbalanced [38]. Here is just a gentle reminder that in our problem, we should handle complex data that usually show unbalancing distributions in Section 3. To measure the precision and recall values of a classification method, we first consider a confusion matrix of classes $M_{i,j}$, where each row of the confusion matrix represents predicted class, while each column represents actual class. n is the number of classes. True positive, False positive, and False negative in each class are represented as Eq. (2).

$$\begin{aligned} \text{Truepositivie}_i &= M_{i,i} \\ \text{Falsepositivie}_i &= \sum\nolimits_{k=1}^{n} M_{i,k} | k \neq i \\ \text{Falsenegative}_i &= \sum\nolimits_{k=1}^{n} M_{k,i} | k \neq i \end{aligned} \tag{2}$$

Based on Eq. (2), the precision, recall, and F_1–score (Harmonic mean between the precision and the recall) are defined as:

$$\begin{aligned} \text{Precision} &= \sum\nolimits_{k=1}^{n} \frac{\text{True positivie}_i}{\text{True positivie}_i + \text{False positivie}_i} \\ \text{Recall} &= \sum\nolimits_{k=1}^{n} \frac{\text{True positivie}_i}{\text{True positivie}_i + \text{False negative}_i} \\ F_1\text{−score} &= \frac{2 \times \text{Precision} \times \text{Recall}}{\{\text{Precision} + \text{Recall}\}} \end{aligned} \tag{3}$$

ROUGE-1 metric refers to normalizing the number of the overlapping words between the system and reference summaries. The reference summary is a set of words made by a domain expert, while the system summary is a set of words generated by a given model. The precision, recall, and F_1–score metrics based on ROUGE-1 are defined as Eq. (4). In the text summarization problem, ROUGE–1∼N metrics are widely used, but we just use the ROUGE–1 metric rather than the other ROUGE metrics because it measures the intersection of the words between system and reference summaries. However, ROUGE–2∼N metrics consider the intersection of the sentences between both summaries. In practice, it is time–consuming and subjective for human experts to manually summarize the content of many web pages. Therefore, in our context, because it is much easier to just consider the intersection of words in both summaries, we used the ROUGE–1 metric to validate the effectiveness of the proposed method.

$$\begin{aligned} &\text{Precision}_{(\text{ROUGE}-1)} \\ &\quad = \frac{\text{\# of overlapping words}}{\text{Total \# of the ords in reference summary}} \\ &\text{Recall}_{(\text{ROUGE}-1)} \\ &\quad = \frac{\text{\# of overlapping words}}{\text{Total \# of the words in the system summary}} \\ &F_1\text{−score}_{(\text{ROUGE}-1)} \\ &\quad = \frac{2 \times \text{Precision}_{\text{ROUGE}-1} \times \text{Recall}_{\text{ROUGE}-1}}{\{\text{Precision}_{\text{ROUGE}-1} + \text{Recall}_{\text{ROUGE}-1}\}} \end{aligned} \tag{4}$$

VI. EXPERIMENTAL RESULTS

A. RESULTS OF MOVIE AND CELLPHONE DATA SETS

In this section, to validate the effectiveness of the proposed algorithm that improves the existing deep learning models, we evaluate 12 classification methods with/without HiTGen in the movie data set. There are 30 movie entities, each of which has eight attributes i.e., $D = \{e_1, e_2, \ldots, e_{30}\}$, where $e_i (\in D) = \{a_1, a_2, \ldots, a_8\}$. Attributes $a_1 \sim a_8$ are Movie name, Director, Actor, Nationality, Genre, Rating level, Running time, and Year, respectively. For example, a_1 = "Iron Man 3", a_2 = "Shane Black", a_3 = "Robert Downey Jr.", a_4 = "US", a_5 = "Action", a_6 = "12 years old", a_7 = "129 minutes", and a_8 = "2013". Because every entity has eight attributes, the total number of queries created by combination of the attributes is $2^8 - 1 = 255$. For every query $q_i \in \{q_1, \ldots, q_{255}\}$, top–10 web pages are retrieved through Google search engine. The reason why we select only top–10 web pages that are highly ranked by the PageRank algorithm of Google is because of the previous study in which

[5]https://www.tensorflow.org
[6]http://www.cs.cornell.edu/people/tj/svm_light/index.html
[7]https://scikit-learn.org

TABLE 5. Average precision, recall, and F_1–score of movie and cellphone data sets. Due to space limitation, we just show the average precision, recall, and F_1–score of all entities per model. Please see Appendix to see the result of each entity in detail.

Data sets	Methods	Precision	Recall	F_1–score
Movies	SVM	0.5237	0.9453	0.6603
	RF	0.7573	0.7866	0.6929
	AB	0.4784	0.4476	0.3825
	FNN	0.6372	0.9701	0.7383
	CNN	0.6722	0.8910	0.7237
	Bi-LSTM	0.7499	0.8945	0.7751
	SVM using HiTGen	0.7092	0.9002	0.7640
	RF using HiTGen	0.9900	0.9897	0.9893
	AB using HiTGen	0.9862	0.9948	0.9900
	FNN using HiTGen	0.9838	0.9858	0.9807
	CNN using HiTGen	0.8663	0.9340	0.8738
	Bi-LSTM using HiTGen	**0.9926**	**0.9910**	**0.9972**
Cellphones	SVM	0.4002	0.9273	0.5579
	RF	0.7972	0.7366	0.6843
	AB	0.4153	0.4858	0.3755
	FNN	0.6682	0.8620	0.6877
	CNN	0.6615	0.8203	0.6649
	Bi-LSTM	0.7625	0.7917	0.7412
	SVM using HiTGen	0.8794	0.7029	0.7588
	RF using HiTGen	**0.9880**	0.9897	0.9426
	AB using HiTGen	0.9871	0.9968	**0.9918**
	FNN using HiTGen	0.9844	0.9792	0.9771
	CNN using HiTGen	0.9219	0.9037	0.8851
	Bi-LSTM using HiTGen	0.9844	**1.0000**	0.9896

a majority of people only look at the first returned page (i.e., 10 links) of Google [39]. When we manually investigated all web pages in Table 3, the ratio of relevant web pages to irrelevant ones is 43% to 57%. In general, movie names are ambiguous. A movie "Chinatown" indicates different meanings – e.g., a recently released film, local restaurants, and foreign districts in big cities. Furthermore, each movie entity shows a large number of clusters and the unbalancing distribution of cluster sizes. To make matters worse, classifying web pages to either relevant or irrelevant suffers from various reasons such as ad and comparison with other entities in a web page. Interestingly, movie names are more ambiguous because they are often named using common words which may be even short to let people remember for a long time.

Table 5 summarizes the average F_1–scores of 30 movie entities. The average F_1–scores of all AI–based classification methods using HiTGen are much better than those of the other classification methods. For example, without HiTGen, RF is the best in the conventional learning models and Bi–LSTM is the best in the deep learning models. The F_1–scores of RF and Bi–LSTM are 0.6929 and 0.7751, respectively, while that of Bi–LSTM using HiTGen is 0.9972. Overall, the recall values are relatively higher than the precision values. The recall value is the fraction of relevant web pages that have been retrieved in the entire solution set. On the other hand, the precision value is the fraction of relevant web pages predicted by a given method. Therefore, in our context, the precision value is more important than the recall value. If the precision value is low, it indicates that the performance of the given method is poor. Please note that the precision values of the conventional and AI–based classification methods are considerably low. However, the proposed algorithm improves the precision values of all methods considerably. For instance, FNN using HiTGen improves about 54% more than the existing FNN. CNN using HiTGen also improves about 29% more than the existing CNN. Bi–LSTM using HiTGen improves about 32% more than the existing Bi–LSTM. The reason why HiTGen improves the existing deep learning models is that it is likely to generate the most discriminative feature vectors. Please recall that HiTGen identifies a set of main words *relevant* with a target entity e. This word set is used as features to make a input vector per web page. Through the experimental results, we observed that relevant and irrelevant input vectors are disjoint in most cases. Meanwhile, the recall values across all methods are consistent. Apparently, the precision values of all methods are consistent as well. However, AB without HiTGen is the worst among all classification methods and its F_1–score is 0.3825. In general, it is known that AB is used in combination with several weak learners and the final learner classifies using the weighted average output of the other learners. AB as one of ensemble methods shows high performance in most cases, but it does not work effectively in complex data including many outliers and noises. It is also relatively vulnerable, compared to the other classification methods, in the overfitting problem. Our experimental results indirectly explain that both Movie and Cellphone data sets are so complex that it is difficult to classify correctly when AB is used. Since such complex data can be represented as a non–linear classification problem, the AI–based methods are much better than the conventional classification methods.

The reason why the deep learning models using the proposed method are better than the existing deep learning models is that the proposed method can generate input vectors with the most discriminative word features. In particular, according to our careful investigation, in the proposed word features (Line 20 in Algorithm 1), the back sequence is relatively less relevant to the entity than the previous sequence. It turns out that the order of the word features (Line 18 and 19 in Algorithm 1) can increase the F_1–score of the existing Bi–LSTM. In addition, max–pooling layers for the input vectors with relevance labels and min–pooling layers for the input vectors with irrelevance labels can improve the F_1–score of the existing CNN models.

We further apply HiTGen to even the traditional classification models such as SVM, RF, and AB. Like the outcome of existing deep learning–based methods using HiTGen, the conventional methods with SVM, RF, and AB using HiTGen show good results as well. For instance, SVM using HiTGen improves about 35% more than the existing SVM. RF using HiTGen improves about 30% more than the existing RF. AB using HiTGen improves about 106% more than the existing AB. These results are promising because the proposed HiTGen method can improve both conventional and AI–based models.

Similarly, Table 5 also summarizes the average F_1–scores of 12 models with eight cellphone entities, each of which has eight attributes like Model name(a_1), Maker(a_2), Year(a_3), RAM(a_4), OS(a_5), Weight(a_6), Screen size(a_7), and Battery(a_8), respectively. An example of the cellphone

TABLE 6. Results of ROUGE-1.

Data sets		Precision	Recall	F_1–score
Movies	TF/IDF	0.0757	0.0302	0.0424
	HiTGen	**0.4045**	**0.2019**	**0.2616**
Cellphones	TF/IDF	0.1255	0.0607	0.0815
	HiTGen	**0.3749**	**0.1982**	**0.2585**

entities is a_1 = "iPhone6", a_2 = "Apple Inc.", a_3 = "January 2014", a_4 = "1GB", a_5 = "iOS8", a_6 = "112g", a_7 = "11.9cm", and a_8 = "2915mAh". The results are fairly close to those of the movie data set except that the overall F_1–scores are slightly lower than those of the movie data set. This is because the duration of cellphones is very short and there are a number of web pages dealing with various models with the same name (e.g., Galaxy S5 and Galaxy S6). These characteristics make the classification process more difficult in the cellphone data set than in the moive data set. Unlike the movie data set, the precision values of both AB and RF using HiTGen are slightly higher than Bi–LSTM using HiTGen. It seems that recursive partitioning and pruning of the decision tree–based methods work effectively because Algorithm 1 (Line 21 to 32) generates good sparse vectors based on relevant top words and irrelevant top words. These experimental results demonstrate that HiTGen definitely improves all conventional and AI–based models.

B. VALIDATION OF HiTGen

We first calculated all precision, recall, and F_1–scores of the movie entities to finally obtain the average values. Similarly, we conducted the same process in the cellphone entities. Table 6 shows the ROUGE-1 results. Regardless of movies or cellphones, all F_1–scores are consistent. The results of HiTGen are more close to those of reference summaries (word sets from the movies' story and the cellphones' story). This indicates that the proposed algorithm is likely to generate more discriminative word features than TF/IDF. Even though HiTGen is better than TF/IDF, its values are not high (e.g., the F_1–scores of HiTGen are 0.2616 and 0.2585 in the movies and cellphones data sets.). The reason is that the number of words in s_3 is much larger than the number of words in s_1 and s_2. Note that s_1 and s_2 contain only top-20 words. As a result, the experimental result clarifies that we can automatically acquire a number of high quality training data using Eq. (1).

To automatically collect high–quality training data, we propose HiTGen in Section IV-A. The core of the algorithm is Eq. (1). To see how effective the equation is, we compare it with TF/IDF metric that is widely used in Information Retrieval. Technically, for each movie entity, we select top–20 relevant words retrieved by HiTGen and insert to a set s_1. In the same way, we also choose top–20 words with the highest TF/IDF value and insert to a set s_2. Finally, we collect all unique words from the movie's story on the web and insert to a set s_3. Then, we compute the precision, recall, and

TABLE 7. Results of the five movie data.

Methods		Precision	Recall	F_1–score
Conjuring	Bi–LSTM	0.7625	0.7917	0.7412
	Bi–LSTM using SeqGAN	0.8923	0.7919	0.8156
	Bi–LSTM using HiTGen	1.0	1.0	1.0
Frozen	Bi–LSTM	0.55	0.6667	0.45
	Bi–LSTM using SeqGAN	0.9147	0.75	0.8020
	Bi–LSTM using HiTGen	1.0	1.0	1.0
Gravity	Bi–LSTM	0.625	1.0	0.7639
	Bi–LSTM using SeqGAN	0.7211	0.8833	0.7661
	Bi–LSTM using HiTGen	0.99	1.0	0.9949
Iron Man III	Bi–LSTM	0.8125	0.75	0.7143
	Bi–LSTM using SeqGAN	0.73	0.726	0.6971
	Bi–LSTM using HiTGen	0.9	1.0	0.9473
Turbo	Bi–LSTM	0.9	0.8125	0.8185
	Bi–LSTM using SeqGAN	0.8503	0.75	0.7915
	Bi–LSTM using HiTGen	1.0	0.875	0.9333
Average	Bi–LSTM	**0.73**	**0.8042**	**0.6976**
	Bi–LSTM using SeqGAN	**0.8217**	**0.7802**	**0.7745**
	Bi–LSTM using HiTGen	**0.978**	**0.975**	**0.9751**

F_1–score between two sets based on ROUGE–1 metric. For example, the precision value between s_1 and s_3 is computed by the number of overlapping words divided by the number of total words in s_1. The recall value between s_1 and s_3 is computed by the number of overlapping words divided by the number of total words in s_3. We can also compute both precision and recall values between s_2 and s_3. If precision(s_1, s_3) > precision(s_2, s_3), the top–20 words chosen by HiTGen are more similar to the words from the movie's stroy and vice versa. For example, in a movie "Conjuring" the following top–20 words extracted by HiTGen are as follows:

- **Relevant top–20 words**: year, children, warren, husband, wife, amityville horror, thaw, patrick, wilson, farmiga, amity, lorraine, conjuring, rating, actor, also, demon, horror movie, insidious
- **Irrelevant top–20 words**: hyun bin, calvin, klein, han ji–min, annabelle, mirror, collection, warren, prima, netizen, webtoon, taissa, sulli, ha jung–woo, hide, wallis, farmiga, tiger, dance, motion

Please note the irrelevant word list in which most words are the actor names of another movies, another horror movie (Annabelle), and the majority of terms not related to the movie.

Recently, in order to improve the accuracy of deep learning models, Generative Adversarial Networks (GAN) is proposed as the state–of–the–art method for generating high–quality training data. In the case of natural languages, SeqGAN [9], a variant of GAN, has been used in various domains because the order of words in a sentence is important. Since our proposed method and SeqGAN have similar goals, we first apply SeqGAN to our problem to generate high–quality training data for deep learning models and then compare the results of the proposed method with those of SeqGAN. Five movies ("Conjouring," "Frozen," "Gravity," "Iron Man III," and "Turbo") are randomly selected from a total of 30 movies, and training data are generated using SeqGAN and HiTGen for each movie data. Both methods show similar results for five movies in Table 7. It seems that Bi–LSTM using HiTGen is much better than Bi–LSTM using SeqGAN. In general,

TABLE 8. Results of conventional classification in movies data set.

Movies	Conventional classification								
	SVM			Random Forest			Adaboost		
	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
Miracle in Cell No. 7	0.4664	0.9316	0.6207	0.7775	0.7129	0.6544	0.3817	0.5417	0.4015
The Flu	0.2081	0.7727	0.3188	0.5242	0.6230	0.5180	0.2700	0.5000	0.3503
Cold Eyes	0.5000	1000	0.6664	0.6864	0.8181	0.6457	0.5249	0.6319	0.4150
Gangnam	0.5862	0.9844	0.7300	0.8423	0.8176	0.7852	0.5000	0.2950	0.3688
Frozen	0.2081	0.7727	0.3188	0.6268	0.6104	0.4967	0.5644	0.6003	0.4159
Blood and Ties	0.1751	0.4158	0.2425	0.7085	0.7419	0.6123	0.2939	0.4271	0.3034
The Face Reader	0.4664	0.9316	0.6207	0.7788	0.7812	0.6861	0.3517	0.4687	0.2926
Ode to My Father	0.7200	1.0000	0.8343	0.6705	0.8668	0.6700	0.5250	0.4710	0.4485
Gravity	0.6200	1.0000	0.7604	0.7991	0.8090	0.7118	0.5179	0.4035	0.3853
The Con Artists	0.7200	1.0000	0.8308	0.6875	0.7456	0.6686	0.5469	0.5937	0.4809
Man in Love	0.5900	1.0000	0.7360	0.8229	0.8126	0.7573	0.5218	0.4697	0.3425
Noah	0.6800	1.0000	0.8078	0.8010	0.8014	0.7266	0.3467	0.3438	0.2749
Roaring Currents	0.7200	1.0000	0.8372	0.7708	0.8692	0.7580	0.5750	0.7145	0.5247
Montage	0.5200	1.0000	0.6785	0.7919	0.7751	0.7035	0.5469	0.3517	0.3769
Man on the Edge	0.4700	1.0000	0.6363	0.7886	0.7415	0.6693	0.5000	0.2193	0.3043
Berlin	0.3295	0.7773	0.4609	0.7434	0.7760	0.6744	0.4012	0.4775	0.3575
Big Match	0.7300	1.0000	0.8426	0.7409	0.8875	0.7371	0.5250	0.4771	0.4545
Hide Seek	0.7000	1.0000	0.8210	0.7153	0.8828	0.7006	0.5783	0.7803	0.4918
C'est si bon	0.6800	1.0000	0.8078	0.8125	0.8438	0.7564	0.5208	0.4535	0.4311
Iron Man III	0.4933	1.0000	0.6579	0.8470	0.7625	0.7311	0.5250	0.3512	0.3498
About Time	0.3200	1.0000	0.4848	0.8456	0.7224	0.7017	0.2998	0.4115	0.2948
Very Ordinary Couple	0.5500	1.0000	0.7086	0.7536	0.8009	0.7089	0.5208	0.3797	0.3728
World War Z	0.4300	1.0000	0.5990	0.7740	0.6958	0.6269	0.5500	0.2762	0.3015
Fists of Legend	0.6500	1.0000	0.7874	0.7062	0.9208	0.7436	0.4342	0.4527	0.3579
Detective K: Secret of the Virtuous Widow	0.5500	1.0000	0.7057	0.8292	0.6805	0.7249	0.5000	0.2650	0.3454
The Conjuring	0.4000	1.0000	0.5714	0.7760	0.7926	0.7108	0.4588	0.4782	0.4131
Turbo	0.2081	0.7727	0.3188	0.7738	0.7495	0.6764	0.5495	0.4910	0.4101
The Target	0.7500	1.0000	0.8561	0.7333	0.8885	0.7276	0.5208	0.4795	0.4501
The Pirates	0.6700	1.0000	0.8009	0.7812	0.8446	0.7487	0.5000	0.335	0.4005
Chronicle of a Blood Merchant	0.6000	1.0000	0.7454	0.8095	0.8220	0.7529	0.5000	0.2867	0.3588
Average	0.5237	0.9453	0.6603	0.7573	0.7866	0.6929	0.4784	0.4476	0.3825

TABLE 9. Results of conventional classification using the proposed algorithm in movies data set.

Movies	Conventional classification using HiTGen								
	SVM			Random Forest			Adaboost		
	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
Miracle in Cell No. 7	0.9365	0.6539	0.7633	0.9740	0.9765	0.9741	0.9679	0.9904	0.9782
The Flu	0.8879	0.8125	0.8476	1.0000	0.9904	0.9950	0.9688	1.0000	0.9832
Cold Eyes	0.5200	1.0000	0.6775	0.9896	1.0000	0.9946	0.9844	1.0000	0.9917
Gangnam	0.6700	1.0000	0.7992	1.0000	1.0000	1.0000	0.9875	1.0000	0.9934
Frozen	0.8854	0.4339	0.5731	0.9654	0.9717	0.9661	1.0000	0.9797	0.9894
Blood and Ties	0.8763	0.8141	0.8396	1.0000	1.0000	1.0000	1.0000	0.9864	0.9930
The Face Reader	0.5000	1.0000	0.6655	1.0000	0.9800	0.9896	0.9803	0.9736	0.9761
Ode to My Father	0.6600	1.0000	0.7920	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Gravity	0.5500	1.0000	0.7086	0.9669	0.9815	0.9724	0.9821	1.0000	0.9907
The Con Artists	0.6900	1.0000	0.8161	0.9844	1.0000	0.9917	0.9674	0.9782	0.9712
Man in Love	0.6400	1.0000	0.7801	0.9875	0.9931	0.9899	1.0000	1.0000	1.0000
Noah	0.7750	0.6987	0.7226	1.0000	0.9693	0.9840	0.9387	1.0000	0.9672
Roaring Currents	0.7100	1.0000	0.8293	1.0000	0.9864	0.9930	0.9896	1.0000	0.9946
Montage	0.5200	1.0000	0.6819	0.9917	0.9669	0.9783	0.9844	1.0000	0.9917
Man on the Edge	0.6000	1.0000	0.7472	1.0000	1.0000	1.0000	0.9803	1.0000	0.9897
Berlin	0.8303	0.7527	0.7868	0.9674	0.9782	0.9712	1.0000	1.0000	1.0000
Big Match	0.6100	1.0000	0.7561	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Hide Seek	0.6400	1.0000	0.7790	0.9490	1.0000	0.9731	0.9833	1.0000	0.9911
C'est si bon	0.7300	1.0000	0.8435	1.0000	0.9868	0.9932	1.0000	1.0000	1.0000
Iron Man III	0.8782	0.6325	0.7238	0.9911	0.9886	0.9894	0.9730	0.9815	0.9761
About Time	0.8909	0.7576	0.8143	1.0000	0.9911	0.9954	1.0000	0.9852	0.9924
Very Ordinary Couple	0.6400	1.0000	0.7780	1.0000	1.0000	1.0000	0.9904	1.0000	0.9950
World War Z	0.4800	1.0000	0.6385	0.9911	0.9722	0.9797	0.9779	0.9911	0.9838
Fists of Legend	0.5800	1.0000	0.7304	0.9730	0.9815	0.9761	1.0000	1.0000	1.0000
Detective K: Secret of the Virtuous Widow	0.6400	1.0000	0.7787	1.0000	0.9852	0.9924	1.0000	1.0000	1.0000
The Conjuring	1.0000	0.6510	0.7851	0.9904	1.0000	0.9950	0.9500	1.0000	0.9729
Turbo	0.8961	0.7994	0.8439	0.9779	0.9911	0.9838	1.0000	1.0000	1.0000
The Target	0.6400	1.0000	0.7779	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
The Pirates	0.7600	1.0000	0.8615	1.0000	1.0000	1.0000	0.9815	0.9782	0.9790
Chronicle of a Blood Merchant	0.6400	1.0000	0.7794	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Average	0.7092	0.9002	0.7640	0.9900	0.9897	0.9893	0.9862	0.9948	0.9900

an ambiguous entity is related to various web pages. For example, an entity ''Troy'' is a set of web pages about a movie, a city, and a university. This data set has a large number of clusters such as the movie cluster, the city cluster, and the university cluster. The number of web pages in each cluster is unbalanced. In addition, in case that the number of web pages is small, the size of training data generated by SeqGAN is small. These reasons tend to prevent SeqGAN from generating high–quality training data. On the other hand, the F_1–score of the proposed method is high because it uses discriminative word features for generating vectors in the training data, where each word comes from high frequent

TABLE 10. Results of AI-based classification in movies data set.

Movies	AI-based classification								
	FNN			CNN			Bi-LSTM		
	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
Miracle in Cell No. 7	0.7292	1.0000	0.8000	0.6250	0.5417	0.4917	0.8750	0.7500	0.7500
The Flu	0.6750	1.0000	0.7937	0.7500	0.4792	0.5333	0.8250	0.6667	0.6722
Cold Eyes	0.5750	1.0000	0.7192	0.7417	0.9167	0.8056	0.8542	0.8750	0.8571
Gangnam	0.7917	1.0000	0.8667	0.6000	1.0000	0.7302	0.8250	1.0000	0.8889
Frozen	0.4500	1.0000	0.6042	0.6000	0.9167	0.6845	0.5500	0.6667	0.4500
Blood and Ties	0.6250	0.5000	0.4167	0.6250	0.7500	0.5833	0.8750	1.0000	0.9167
The Face Reader	0.6250	1.0000	0.7639	0.7875	0.9375	0.8214	0.6250	1.0000	0.7639
Ode to My Father	0.6500	1.0000	0.7827	0.6500	1.0000	0.7827	0.8000	1.0000	0.8750
Gravity	0.6833	0.9167	0.7764	0.7000	1.0000	0.8026	0.6250	1.0000	0.7639
The Con Artists	0.7250	1.0000	0.8214	0.6500	1.0000	0.7679	0.7500	1.0000	0.8304
Man in Love	0.6375	1.0000	0.7669	0.6250	1.0000	0.7639	0.6833	0.9167	0.7764
Noah	0.6500	0.7500	0.6042	0.5417	0.6250	0.4917	0.6875	0.6250	0.5476
Roaring Currents	0.7000	1.0000	0.8194	0.7000	1.0000	0.8194	0.7375	1.0000	0.8462
Montage	0.6500	1.0000	0.7748	0.7167	1.0000	0.8319	0.8500	0.8125	0.7907
Man on the Edge	0.5750	0.9375	0.6667	0.7917	0.9375	0.8310	0.7000	1.0000	0.8056
Berlin	0.7250	1.0000	0.8383	0.7500	0.6875	0.7143	0.8750	0.7500	0.8000
Big Match	0.7625	1.0000	0.8651	0.6500	1.0000	0.7847	0.7250	1.0000	0.8383
Hide Seek	0.4750	1.0000	0.6399	0.4750	1.0000	0.6399	0.6667	0.8750	0.6970
C'est si bon	0.6500	1.0000	0.7847	0.6500	1.0000	0.7847	0.6500	1.0000	0.7847
Iron Man III	0.4750	1.0000	0.6097	0.6458	0.9375	0.7292	0.8125	0.7500	0.7143
About Time	0.4292	1.0000	0.5708	0.5625	0.7917	0.6167	0.7500	0.6667	0.6167
Very Ordinary Couple	0.7292	1.0000	0.8310	0.6375	1.0000	0.7500	0.7875	1.0000	0.8571
World War Z	0.5542	1.0000	0.6889	0.7500	0.8750	0.7500	0.8667	1.0000	0.9222
Fists of Legend	0.6500	1.0000	0.7679	0.7125	1.0000	0.8185	0.7500	1.0000	0.8304
Detective K: Secret of the Virtuous Widow	0.5625	1.0000	0.6556	0.5000	1.0000	0.6111	0.8250	1.0000	0.8889
The Conjuring	0.5875	1.0000	0.7321	1.0000	0.6250	0.7417	0.6250	0.6667	0.6286
Turbo	0.7000	1.0000	0.8026	0.7917	0.7083	0.7429	0.9000	0.8125	0.8185
The Target	0.6625	1.0000	0.7907	0.6000	1.0000	0.7401	0.6000	1.0000	0.7401
The Pirates	0.6875	1.0000	0.7698	0.6500	1.0000	0.7431	0.7500	1.0000	0.8056
Chronicle of a Blood Merchant	0.7250	1.0000	0.8264	0.6875	1.0000	0.8016	0.6500	1.0000	0.7748
Average	0.6372	0.9701	0.7383	0.6722	0.8910	0.7237	0.7499	0.8945	0.7751

TABLE 11. Results of AI-based classification using the proposed algorithm in movies data set.

Movies	AI-based classification using HiTGen								
	FNN			CNN			Bi-LSTM		
	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
Miracle in Cell No. 7	0.8750	1.0000	0.9167	0.8750	1.0000	0.9167	0.9900	1.0000	1.0000
The Flu	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9900	1.0000	1.0000
Cold Eyes	1.0000	1.0000	1.0000	0.9375	1.0000	0.9643	1.0000	1.0000	1.0000
Gangnam	1.0000	1.0000	1.0000	0.9500	1.0000	0.9722	1.0000	1.0000	1.0000
Frozen	1.0000	1.0000	1.0000	0.9000	0.8125	0.8185	0.9975	1.0000	1.0000
Blood and Ties	1.0000	1.0000	1.0000	1.0000	0.9167	0.9500	1.0000	1.0000	1.0000
The Face Reader	1.0000	1.0000	1.0000	0.9167	1.0000	0.9500	0.9800	1.0000	1.0000
Ode to My Father	1.0000	1.0000	1.0000	0.8542	0.9375	0.8786	0.9900	1.0000	1.0000
Gravity	1.0000	1.0000	1.0000	0.9167	1.0000	0.9500	0.9900	1.0000	1.0000
The Con Artists	1.0000	0.9500	0.9722	0.7667	1.0000	0.8429	0.9900	1.0000	1.0000
Man in Love	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Noah	1.0000	1.0000	1.0000	0.7500	0.6250	0.6250	0.9700	0.9167	0.9167
Roaring Currents	1.0000	1.0000	1.0000	0.9000	1.0000	0.9444	1.0000	1.0000	1.0000
Montage	0.9500	1.0000	0.9722	0.9500	0.9375	0.9365	0.9800	1.0000	1.0000
Man on the Edge	1.0000	1.0000	1.0000	0.8000	1.0000	0.8750	0.9900	1.0000	1.0000
Berlin	1.0000	1.0000	1.0000	0.8750	0.7500	0.7500	1.0000	1.0000	1.0000
Big Match	1.0000	1.0000	1.0000	0.8500	1.0000	0.9097	0.9800	1.0000	1.0000
Hide Seek	0.8125	0.6875	0.6786	0.7375	1.0000	0.7976	0.9900	0.9375	1.0000
C'est si bon	1.0000	1.0000	1.0000	0.8875	1.0000	0.9365	1.0000	1.0000	1.0000
Iron Man III	1.0000	1.0000	1.0000	0.9000	1.0000	0.9375	0.9900	1.0000	1.0000
About Time	1.0000	1.0000	1.0000	0.8750	0.6667	0.7000	0.9900	1.0000	1.0000
Very Ordinary Couple	1.0000	1.0000	1.0000	0.6875	1.0000	0.7946	0.9900	1.0000	1.0000
World War Z	1.0000	1.0000	1.0000	0.7500	0.6250	0.6250	1.0000	1.0000	1.0000
Fists of Legend	1.0000	1.0000	1.0000	0.8667	1.0000	0.9222	1.0000	1.0000	1.0000
Detective K: Secret of the Virtuous Widow	1.0000	1.0000	1.0000	0.7875	1.0000	0.8571	0.9900	1.0000	1.0000
The Conjuring	1.0000	1.0000	1.0000	0.8750	0.7500	0.7500	1.0000	1.0000	1.0000
Turbo	0.8750	0.9375	0.8810	0.8750	1.0000	0.9167	1.0000	0.8750	1.0000
The Target	1.0000	1.0000	1.0000	0.8167	1.0000	0.8875	1.0000	1.0000	1.0000
The Pirates	1.0000	1.0000	1.0000	0.7375	1.0000	0.8294	1.0000	1.0000	1.0000
Chronicle of a Blood Merchant	1.0000	1.0000	1.0000	0.9500	1.0000	0.9722	0.9800	1.0000	1.0000
Average	0.9838	0.9858	0.9807	0.8663	0.9340	0.8738	0.9926	0.9910	0.9972

web pages that are considered as web pages relevant with an entity of interest.

VII. DISCUSSION

To classify web pages as relevant or irrelevant with a target entity, based on the proposed method called HiTGen, both traditional classification models (i.e., SVM, RF, and AB) and existing deep learning models (i.e., FNN, CNN, and RNN) improve F_1-scores largely. Moreover, the two experimental data sets (i.e., movies and cellphones) show the same results, which show consistent results regardless of the characteristics of the data set. Through HiTGen, the high-quality

TABLE 12. Results of conventional classification in cellphones data set.

Cellphones	Conventional classification								
	SVM			Random Forest			Adaboost		
	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
iPhone 5	0.4664	0.9316	0.6207	0.7869	0.7865	0.6987	0.4156	0.4418	0.3310
iPhone 6	0.4400	1.0000	0.6111	0.8125	0.7428	0.6987	0.4811	0.6196	0.4774
Galaxy Note 4	0.3366	0.8331	0.4784	0.8068	0.6402	0.6727	0.2968	0.4219	0.3038
Galaxy S5	0.3550	0.8761	0.5046	0.8042	0.7515	0.6971	0.2802	0.4271	0.2886
Galaxy S6	0.3942	1.0000	0.5651	0.9083	0.7024	0.7317	0.6548	0.5217	0.5265
Galaxy S6 Edge	0.3295	0.7773	0.4609	0.7197	0.7958	0.6489	0.4225	0.4789	0.3825
G3	0.4400	1.0000	0.6111	0.7549	0.7560	0.6702	0.3367	0.5179	0.3260
G4	0.4400	1.0000	0.6111	0.7840	0.7172	0.6562	0.4349	0.4571	0.3685
Average	0.4002	0.9273	0.5579	0.7972	0.7366	0.6843	0.4153	0.4858	0.3755

TABLE 13. Results of conventional classification using the proposed algorithm in cellphones data set.

Cellphones	Conventional classification using HiTGen								
	SVM			Random Forest			Adaboost		
	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
iPhone 5	1.0000	0.4678	0.6229	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
iPhone 6	1.0000	0.6132	0.7549	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Galaxy Note 4	0.9167	0.4218	0.5661	0.9917	1.0000	0.9957	0.9917	1.0000	0.9957
Galaxy S5	0.8161	0.9808	0.8907	1.0000	0.9815	0.9904	1.0000	0.9904	0.9950
Galaxy S6	0.8293	0.8637	0.8383	0.9896	1.0000	0.9946	0.9896	1.0000	0.9946
Galaxy S6 Edge	1.0000	0.8981	0.9459	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
G3	0.8277	0.6794	0.7451	0.9535	0.9790	0.6440	0.9541	0.9904	0.9707
G4	0.7660	0.4630	0.5708	0.9812	0.9675	0.9732	0.9743	1.0000	0.9864
Average	0.8794	0.7029	0.7588	0.9880	0.9897	0.9426	0.9871	0.9968	0.9918

TABLE 14. Results of AI-based classification in cellphones data set.

Cellphones	AI-based classification								
	FNN			CNN			Bi-LSTM		
	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
iPhone 5	0.5750	0.7500	0.5365	0.7833	0.7083	0.6389	0.8125	0.6875	0.6786
iPhone 6	0.9500	0.6667	0.7556	0.7500	0.6042	0.6643	0.7750	0.8750	0.8125
Galaxy Note 4	0.7500	0.5625	0.5476	0.5250	1.0000	0.6746	0.6250	0.6250	0.5000
Galaxy S5	0.6875	1.0000	0.8095	0.5500	1.0000	0.7054	1.0000	0.9167	0.9500
Galaxy S6	0.7292	0.9167	0.8071	0.8333	0.5833	0.6167	0.9375	0.9167	0.9143
Galaxy S6 Edge	0.5042	1.0000	0.6405	0.7500	0.6667	0.6583	0.4583	0.6250	0.5167
G3	0.4125	1.0000	0.5583	0.4000	1.0000	0.5417	0.6875	0.8333	0.7310
G4	0.7375	1.0000	0.8462	0.7000	1.0000	0.8194	0.8042	0.8542	0.8264
Average	0.6682	0.8620	0.6877	0.6614	0.8203	0.6649	0.7625	0.7917	0.7412

TABLE 15. Results of AI-based classification using the proposed algorithm in cellphones data set.

Cellphones	AI-based classification using HiTGen								
	FNN			CNN			Bi-LSTM		
	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
iPhone 5	0.8750	1.0000	0.9167	1.0000	0.8750	0.9167	1.0000	1.0000	1.0000
iPhone 6	1.0000	0.9167	0.9500	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Galaxy Note 4	1.0000	0.9167	0.9500	0.7500	0.5000	0.5000	1.0000	1.0000	1.0000
Galaxy S5	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Galaxy S6	1.0000	1.0000	1.0000	1.0000	0.9375	0.9643	1.0000	1.0000	1.0000
Galaxy S6 Edge	1.0000	1.0000	1.0000	0.8750	1.0000	0.9167	1.0000	1.0000	1.0000
G3	1.0000	1.0000	1.0000	1.0000	0.9167	0.9500	1.0000	1.0000	1.0000
G4	1.0000	1.0000	1.0000	0.7500	1.0000	0.8333	0.8750	1.0000	0.9167
Average	0.9844	0.9792	0.9771	0.9219	0.9037	0.8851	0.9844	1.0000	0.9896

training data for the classification models is generated automatically.

In our work, we pay attention to the working hypothesis that a few web pages with high frequency are relatively relevant with a target entity rather than most web pages with low frequency. In our framework, given a pre-defined entity $e = \{a_1, a_2, \ldots, a_n\}$, where a_i is one of the attributes, we create all possible queries using the attribute values of e. For each query, top-k web pages are retrieved through a certain search engine. If a web page w is retrieved by many queries, we consider it to have high frequency. According to our hypothesis, if w has high frequency, it is considered to be relevant with e. This is, the class label of w is relevant. After extracting top-l words that are important and frequently appear in web pages with high frequency, we assign such words to the feature set. With the class label and the feature set, we automatically construct the high-quality training set for the classification models.

In general, many entities are ambiguous and the number of web pages containing a target entity is at most hundreds. Even such web pages are relevant to several other entities rather than the target entity. Furthermore, the numbers of the web pages belonging to those entities are unbalanced. For these reasons, the accuracy of the machine learning-based web page classification models is not high. Unlike the traditional classification models and the existing deep learning models, we attempt to improve the accuracy by improving the quality of the train set. In particular, our proposed method contributes to how high-quality it is while the train set is automatically generated.

VIII. CONCLUDING REMARK AND FUTURE WORK

In this article, we address automatic identification of relevant web pages. While the existing excellent models show poor results (i.e., up to 0.7 in F_1–score in our experiments), to the best of our knowledge, in the web page classification problem, considering the high frequency of the retrieved web pages, it is the first study to propose an automatic algorithm of generating high–quality training data to considerably improve the accuracy of the existing deep learning models. Our experimental results show that plain deep learning models based on the proposed method outperform the best web page classification models.

In the future work, we plan to apply the proposed model to various domains and we will further develop a web–based prototype system for proof–of–concept.

APPENDIX A
THE DETAILED EXPERIMENTAL RESULTS

See Tables 8–15.

REFERENCES

[1] C. Zhai and S. Massung, *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. New York, NY, USA: ACM, 2016.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (Adaptive Computation and Machine Learning Series). Cambridge, MA, USA: MIT Press, Nov. 2016.

[3] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer, 2010.

[4] T. Hastie, J. Friedman, and R. Tisbshirani, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY, USA: Springer, 2017.

[5] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[6] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[7] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 98–113, Jan. 1997.

[8] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," 2015, *arXiv:1508.01991*. [Online]. Available: http://arxiv.org/abs/1508.01991

[9] L. Yu, W. Zhang, J. Wang, and Y. S. Yu, "Sequence generative adversarial nets with policy gradient. 489 in," in *Proc. AAAI Conf. Artif. Intell.*, vol. 490, 2017, pp. 2852–2858.

[10] I. Bounhas, N. Soudani, and Y. Slimani, "Building a morpho-semantic knowledge graph for Arabic information retrieval," *Inf. Process. Manage.*, vol. 57, no. 6, Nov. 2020, Art. no. 102124.

[11] Y. Djenouri, A. Belhadi, P. Fournier-Viger, and J. C.-W. Lin, "Fast and effective cluster-based information retrieval using frequent closed itemsets," *Inf. Sci.*, vol. 453, pp. 154–167, Jul. 2018.

[12] J. Wang, M. Pan, T. He, X. Huang, X. Wang, and X. Tu, "A pseudo-relevance feedback framework combining relevance matching and semantic matching for information retrieval," *Inf. Process. Manage.*, vol. 57, no. 6, Nov. 2020, Art. no. 102342.

[13] F. Ensan and F. Al-Obeidat, "Relevance-based entity selection for ad hoc retrieval," *Inf. Process. Manage.*, vol. 56, no. 5, pp. 1645–1666, Sep. 2019.

[14] F. Chen, Y. Liu, J. Li, M. Zhang, and S. Ma, "A pruning algorithm for optimal diversified search," in *Proc. 23rd Int. Conf. World Wide Web*, 2014, pp. 237–238.

[15] L. Dietz, "ENT rank: Retrieving entities for topical information needs through entity-neighbor-text relations," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2019, pp. 215–224.

[16] F. Lashkari, E. Bagheri, and A. A. Ghorbani, "Neural embedding-based indices for semantic search," *Inf. Process. Manage.*, vol. 56, no. 3, pp. 733–755, May 2019.

[17] G. Madjarov, V. Vidulin, I. Dimitrovski, and D. Kocev, "Web genre classification with methods for structured output prediction," *Inf. Sci.*, vol. 503, pp. 551–573, Nov. 2019.

[18] K. Zielinski, R. Nielek, A. Wierzbicki, and A. Jatowt, "Computing controversy: Formal model and algorithms for detecting controversy on wikipedia and in search queries," *Inf. Process. Manage.*, vol. 54, no. 1, pp. 14–36, Jan. 2018.

[19] M. Jabreel and A. Moreno, "A deep learning-based approach for multi-label emotion classification in tweets," *Appl. Sci.*, vol. 9, no. 6, p. 1123, Mar. 2019.

[20] A. Bondielli and F. Marcelloni, "A survey on fake news and rumour detection techniques," *Inf. Sci.*, vol. 497, pp. 38–55, Sep. 2019.

[21] P. Yang, G. Zhao, and P. Zeng, "Phishing website detection based on multidimensional features driven by deep learning," *IEEE Access*, vol. 7, pp. 15196–15209, 2019.

[22] W. Zhang, Y. Du, T. Yoshida, and Q. Wang, "DRI-RCNN: An approach to deceptive review identification using recurrent convolutional neural network," *Inf. Process. Manage.*, vol. 54, no. 4, pp. 576–592, Jul. 2018.

[23] R. Barbado, O. Araque, and C. A. Iglesias, "A framework for fake review detection in online consumer electronics retailers," *Inf. Process. Manage.*, vol. 56, no. 4, pp. 1234–1244, Jul. 2019.

[24] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma, "Multi-label learning with millions of labels: Recommending advertiser bid phrases for Web pages," in *Proc. 22nd Int. Conf. World Wide Web (WWW)*, 2013, pp. 13–24.

[25] D. Eswaran, P. N. Bennett, and J. J. Pfeiffer, "Modeling website topic cohesion at scale to improve webpage classification," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2015, pp. 787–790.

[26] O. Wu, R. Hu, X. Mao, and W. Hu, "Quality-based learning for Web data classification," in *Proc. AAAI Conf. Artif. Intell.*, 2014, vol. 28, no. 1, pp. 1–4.

[27] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 7370–7377.

[28] F. S. Al-Anzi and D. AbuZeina, "Beyond vector space model for hierarchical arabic text classification: A Markov chain approach," *Inf. Process. Manage.*, vol. 54, no. 1, pp. 105–115, Jan. 2018.

[29] H. Wang and M. Hong, "Supervised Hebb rule based feature selection for text classification," *Inf. Process. Manage.*, vol. 56, no. 1, pp. 167–191, Jan. 2019.

[30] Y. Wu, S. Zhao, and W. Li, "Phrase2Vec: Phrase embedding based on parsing," *Inf. Sci.*, vol. 517, pp. 100–127, May 2020.

[31] Y. Kim, Y. Jernite, D. Sontag, and A. Rush, "Character-aware neural language models," in *Proc. AAAI Conf. Artif. Intell.*, 2016, vol. 30, no. 1, pp. 1–9.

[32] R. Johnson and T. Zhang, "Supervised and semi-supervised text categorization using LSTM for region embeddings," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 526–534.

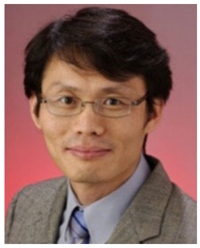
[33] R. You, Z. Zhang, Z. Wang, S. Dai, H. Mamitsuka, and S. Zhu, "AttentionXML: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification," 2018, *arXiv:1811.01727*. [Online]. Available: http://arxiv.org/abs/1811.01727

[34] S. Mukherjee and A. Awadallah, "Uncertainty-aware self-training for few-shot text classification," in *Proc. Adv. Neural Inf. Process. Syst.*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Red Hook, NY, USA: Curran Associates, 2020, pp. 21199–21212.

[35] B. On, M. Omar, G. Choi, and J. Kwon, "Gathering Web pages of entities with high precision," *J. Web Eng.*, vol. 13, nos. 5–6, pp. 378–404, 2014.
[36] S. K. Lee, B.-W. On, and S.-M. Jung, "Mining search keywords for improving the accuracy of entity search," *KIPS Trans. Softw. Data Eng.*, vol. 5, no. 9, pp. 451–464, Sep. 2016.
[37] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks Trade*. Berlin, Germany: Springer, 2012, pp. 9–48.
[38] F. Provost and T. Fawcett, *Data Science for Business: What you Need to Know About Data Mining and Data-Analytic Thinking*. Sebastopol, CA, USA: O'Reilly Media, 2013.
[39] B. J. Jansen and A. Spink, "An analysis of Web documents retrieved and viewed," in *Proc. Int. Conf. Internet Comput.*, 2003, pp. 65–69.

JEONG-JAE KIM is currently pursuing the master's degree in cognitive science with Yonsei University, Seoul, South Korea. His research interests include data mining, artificial intelligence, reinforcement learning, mainly working on abstractive summarization, explainable deep learning, and neuroscience.

BYUNG-WON ON received the Ph.D. degree from the Department of Computer Science and Engineering, The Pennsylvania State University at University Park, PA, USA, in 2007. For several years, he worked as a full-time Researcher with The University of British Columbia and Advanced Institution of Convergence Technology. Since 2014, he has been a Faculty Member with the Department of Software Convergence Engineering, Kunsan National University, South Korea. His recent research interests include data mining, especially probability theory and applications, and artificial intelligence, mainly working on abstractive summarization, creative computing, and multi-agent reinforcement learning.



INGYU LEE received the Ph.D. degree from the Department of Computer Science and Engineering, Pennsylvania State University, with a focus on scientific computing algorithm and software. He is currently an Associate Professor with Sorrell College of Business, Troy University. His main research interests include developing efficient scientific computing algorithms based on mathematical modeling using high performance computing architectures and their applications to real world problems, including information retrieval, data mining, and social networking.