A Topic-Specific Web Crawler using Deep Convolutional Networks

Saed ALqaraleh
Department of Computer Engineering,
Hasan Kalyoncu University,Turkey
saed.alqaraleh@hku.edu.tr

Hatice Meltem Nergız Sırın
Software Engineering Department,
Hasan Kalyoncu University,Turkey
hatice.sirin@hku.edu.tr

Abstract: *This paper presented a new focused crawler that efficiently supports the Turkish language. The developed architecture was divided into multiple units: a control unit, crawler unit, link extractor unit, link sorter unit, and natural language processing unit. The crawler's units can work in parallel to process the massive amount of published websites. Also, the proposed Convolutional Neural Network (CNN) based natural language processing unit can professionally classifying Turkish text and web pages. Extensive experiments using three datasets have been performed to illustrate the performance of the developed approach. The first dataset contains 50,000 Turkish web pages downloaded by the developed crawler, while the other two are publicly available and consist of "28,567" and "22,431" Turkish web pages, respectively. In addition, the Vector Space Model (VSM) in general and word embedding state-of-the-art techniques, in particular, were investigated to find the most suitable one for the Turkish language. Overall, results indicated that the developed approach had achieved good performance, robustness, and stability when processing the Turkish language. Also, Bidirectional Encoder Representations from Transformer (BERT) was found to be the most appropriate embedding for building an efficient Turkish language classification system. Finally, our experiments showed superior performance of the developed natural language processing unit against seven state-of-the-art CNN classification systems. Where accuracy improvement compared to the second-best is 10% and 47% compared to the lowest performance.*

Keywords: *CNN, natural language processing, text classification, topic specific crawler, focused crawler, web crawling.*

Received March 27, 2021; accepted October 26, 2022
https://doi.org/10.34028/iajit/20/3/3

1. Introduction

The World Wide Web (WWW or Web for short) is an information system consisting of interconnected hypertext documents published on the Internet. In general, search engines are systems, particularly websites, that aim to help users find the most accurate results quickly and easily without getting lost/stuck in the almost infinite world of the Internet. Briefly, the main components of any search engine are the user interface, the query engine (including some sub-components such as the matcher and sorter), the indexer, the crawler, and the repository [24, 25]. Users write the query using the search engine's interface to access information on the web. Then, the query engine, which uses some query processing and ranking algorithms, shows a list of results with the most relevant candidates at the top of the list. Hence, these two processes are known as online processes.

On the other hand, offline processes start by collecting the website's information which is done by software known as a crawler, followed by the process of indexing the collected information. Next, search engines build/update their databases (Repository) with the help of web crawlers. The repository manages and stores the website's contents, such as the Hyper Text Markup Language (HTML) web pages, metadata, multimedia files, etc.

As this work is related to the web crawler, its information and details are summarized below. The web contains a massive amount of published websites, around 1.5 billion working ones, and numerous web pages. In addition, the contents of most of those websites are very frequently updated. Search engines work on automatically indexing the website's information as much and as fast as possible. This process is done by traversing the internet and gathering information through software known as crawlers or spiders, or bots. Hence, crawlers visit the indexed sites regularly to add/update their content as well.

Web crawlers generally consist of three sub-modules: the downloader, the extractor, and the sorter. The downloader starts to browse the Uniform Resource Locators (URLs) in its list (seeds). Then, it fetches and downloads a copy of the documents. However, with the vast amount of available web pages, most existing crawlers only download the most important pages of each visited website [18, 21]. On the other hand, the extractor works on detecting and extracting the URLs from the recently downloaded files. These files are then submitted to the search engine indexer to be added to the repository, whereas the extracted URLs are sent to the crawler sorter. The sorter work on deleting the duplicate URLs and then adding the unique ones into the appropriate position in the frontier (list of the extracted URLs), i.e., the list of websites to be visited. Hence, many factors, such as the importance of the website, backlink count, frequency of updating the website content, etc., are used in the ranking process. The entire crawling process is repeated until no more URLs are left, or the target number of web documents has been downloaded.

On the other hand, with the recent development of deep learning, the accuracy of web page classification was significantly improved. However, it is still challenging to classify web pages compared to other types of documents due to:

1. The wide variety of noisy information and semi-structured data embedded in the web page.
2. The problems of constantly/routinely adding new content to web pages.
3. Webpages may contain content that belongs to multiple subjects (topics).
4. The lack of descriptive data sets can be used to build and train efficient classification models.

2. Related Work

Focused or topic-specific crawling usually involves collecting documents about a specific topic(s) [17, 20]. These systems are not new paradigms in the literature but must provide a new solution for growing data volume, indexing velocity, and variety. The research of [6] emphasized that it is a challenge for users to find and visit only the related pages of the web because of the substantial volume of data. Next, we summarize the information on recently developed topic-specific crawlers. A Focused crawler named query-based crawler was introduced in [15]. This crawler works based on a set of keywords set by the user, which is used to

1. Find some relevant webpages using the search option of the visited website (if available).
2. Use google API to find the first ten relevant results. Then, the results of the two methods are ranked and merged into the results list [15].

Another focused crawler presented in [2] uses page weight for the downloaded pages to improve the order of the extracted URLs in the hope of visiting the most significant websites first [2].

Sekhar *et al*. [22] presents a focused crawler Master-Slave architecture for bioinformatics web resources using the Natural Language Processing (NLP) concept where the slaves are the crawlers and coordinates are the master. The master overtakes multiple kernel threads where each is running a socket handler. On the other hand, the slave performs three specific tasks, i.e., downloading the webpage, extracting content, and calculating the relevance score. Although the approach of [22] offers a scalable architecture that reduces the operating time of crawling by increasing the number of parallelized multi-slaves, it requires extra recourses such as Central Processing Unit (CPU) and main memory, and the execution time of calculating the relevance score is significant.

Having an efficient classifier is essential for topic-specific crawling. To validate the proposed classifier, presented in the following section, we compare its performance against seven state-of-the-art CNN models developed for the same domain of our work, i.e., the purpose of text classification, and their details are summarized below.

Demirci *et al*. [11], introduced CNN model, “Turkish_CNN,” consisted of 6 dense layers, where the dropout is applied three times, i.e., 0.5 after the second and the fourth layer, while 0.2 is allocated after the third layer. This model was designed for binary Turkish text classification (2 classes). In addition, the Adam optimizer is used as the optimizer, and Binary Cross Entropy was chosen as the loss function.

“Turkish_Rand” is the model of [16], a smaller version of the original rand CNN network. This model was produced by eliminating some filters and reducing the sizes of other filters and layers. This model starts with an embedding layer followed by a dropout. Then two independent blocks of a convolution, a max-pooling and a flatten layers, are used. These two blacks are merged using a concatenation layer. Finally, another dropout layer is added and followed by two dense layers.

The third model, known as “One layer” [14], its name referred to the fact that this model has one convolution layer. The structure of this model consisted of an embedding layer followed by a convolution layer, then a max-pooling followed by one fully connected layer (dense).

“SensitivityAnalysis” [28] is the name of the fourth model. It uses three convolutions blocks of sizes 2, 3, and 4, where each one has two filters. Then 1-max pooling is integrated. Finally, a Softmax layer is used as the model classifier.

Another popular model is the “VdCNN” [10]: A very deep CNN model consisting of 29 layers of convolution blocks. Each block has two convolutional layers followed by a batch normalization layer and ReLU activation.

Also, “CharCNN” or the “9-layer model” [27], was developed for text classification and consisted of six convolution layers followed by three fully connected layers.

The seventh model is the “CharTCN” [7], i.e., temporal convolutional network (TCN), this approach produces an output of the same length as the input by using one Dimension (1D) fully convolutional where each hidden layer is the same length as the input layer, and zero paddings are used whenever needed. In addition, TCN uses causal convolutions, where an output of the current time is convolved only with elements from the current and the earlier times. Hence, causal convolutions were used to ensure no leakage from the future into the past.

It is worth mentioning that five models of the above ones were initially built for the English language. Unfortunately, based on our investigation, very few studies, such as the ones of [9, 11, 16, 26], have worked on using CNN models for the Turkish language. In addition, out of the mentioned Turkish studies, we found only two of these models [11, 16], where their structure and details can be found in the related references and can be reimplemented.

When it is come to classifying webpages in general, and the Turkish ones in particular, it is essential to have a classifier that can efficiently process the downloaded pages based on their content and metadata [1, 4, 5, 18, 19, 23, 29]. The most similar and relative state-of-the-art classification approaches related to the proposed one are summarized below. In [18], web pages were classified using semantic and structure joint features (meaning, structure, semi-structured data, and HTML syntax). This approach uses Word2vec and Skip-Gram models to estimate the probability of the word in the ranges of its front and back windows based on the probability of the current target word. Two models were used in the experiments of this study, Joint Features Convolutional Neural Networks (JFCNN) and Text Features Convolutional Neural Networks (TFCNN). As a result, JFCNN improved the performance in the range of [4%- 6%].

In [29], a CNN and RNN-based composite feature extraction network, which obtains high dimensional features, including title, text content, and web page description, was introduced. Here, a bidirectional RNN was used for classifying short text such as title and description. On the other hand, the CNN model, which uses both word and paragraph-level, was designed to analyze the long text (the web page's content). Experiments have shown that the module of [29] has an efficient anti-noise feature extractor and achieved good classification accuracy.

The motivations for carrying out this study are

1. The lack of web crawlers that support the Turkish language. Introducing such tools can improve the overall performance of Turkish language search engines. Also, it can be used for building some useful research datasets, and
2. The gap between the current improvements of the CNN-based classifier when processing text, especially in languages such as English, where it can achieve human expert level. While studies on agglutinative languages such as Turkish are limited and at a preliminary stage. We aim to improve the performance and mechanism of Turkish text classification.

The main contributions of this work are:

1. Introducing the first-ever focused crawler that can support the Turkish language.
2. Introducing an efficient CNN-based classifier that was developed after intensive investigation to build the most appropriate one. This classifier can efficiently categorize Turkish text and web pages.
3. To obtain the best performance and to be able to process the vast number of published websites, parallel principles were used by developing an architecture that is divided into multiple units that work in parallel.
4. Investigate state-of-the-art embedding approaches such as Bidirectional Encoder Representations from Transformer (BERT), Elmo, and XLNet to find the most suitable one for the Turkish language.

3. The Developed Crawler

As shown in Figure 1, to improve the performance of the developed crawler, its architecture was divided into multiple units that can work in parallel. Briefly, the mechanism of the proposed crawler is as follows: Controller unit prepares the list of URLs that will be visited (frontier(s)) and distributes them between the working crawlers, where each crawler unit downloads the visited webpages. The link extractor unit processes the downloaded pages, and all URLs inside these pages are extracted. These extracted URLs will be ranked and added to the appropriate position in the main URL frontier(s) by the link sorter unit. Finally, the natural language processing unit works on classifying the downloaded Turkish web pages using the developed CNN classifier.

Next, we will discuss the units of the developed crawler in detail.

- *Controller unit*: the responsibilities of this unit are:
  1. Determining the number of running crawlers and their associated threads.
  2. Caching Domain Name System (DNS) tables: when the crawler connects to a website, it first communicates with the DNS server, which converts the domain name into an IP address. At this point, the controller unit of the developed crawler is responsible for preparing and maintaining DNS caches. This is due to the fact that connecting to DNS may require a long time. By doing so, caching DNS, i.e., converting the domain name into an IP address, became an offline process. Hence, the crawler’s frontiers contain the IPs instead of the domain names, which will speed up the crawling processes and increase the throughputs.
  3. Distributing the URLs in the main frontier(s) between the running crawlers.
- *Crawler Unit*: multiple copies of this unit can run concurrently. Each copy is responsible for visiting the IPs in its frontier, downloading the visited webpages, and saving them into a directory named

"Last Downloaded Static Pages." Note that to avoid degrading the performance of any visited server, the developed crawler was designed to allow only one copy of the crawler unit to visit the server simultaneously.

- *Link Extractor Unit*: this unit takes the pages from the "LastTurkishDownloadedPages" directory, then extracts all URLs inside these pages and adds them to the "ExtractedURLsQueue." Mainly, the following operations are carried out by this unit:

  a) Detecting the language of the downloaded pages, i.e., pages allocated at the "LastDownloadedStaticPages" directory, then saving the non-Turkish pages in the database while moving the Turkish ones to the "LastTurkishDownloadedPages" directory.

  b) To obey the webmaster restriction, the URLs with the "rel" feature set to "nofollow" are ignored and not added to the queue.

  c) Removing duplicate web pages and URLs.

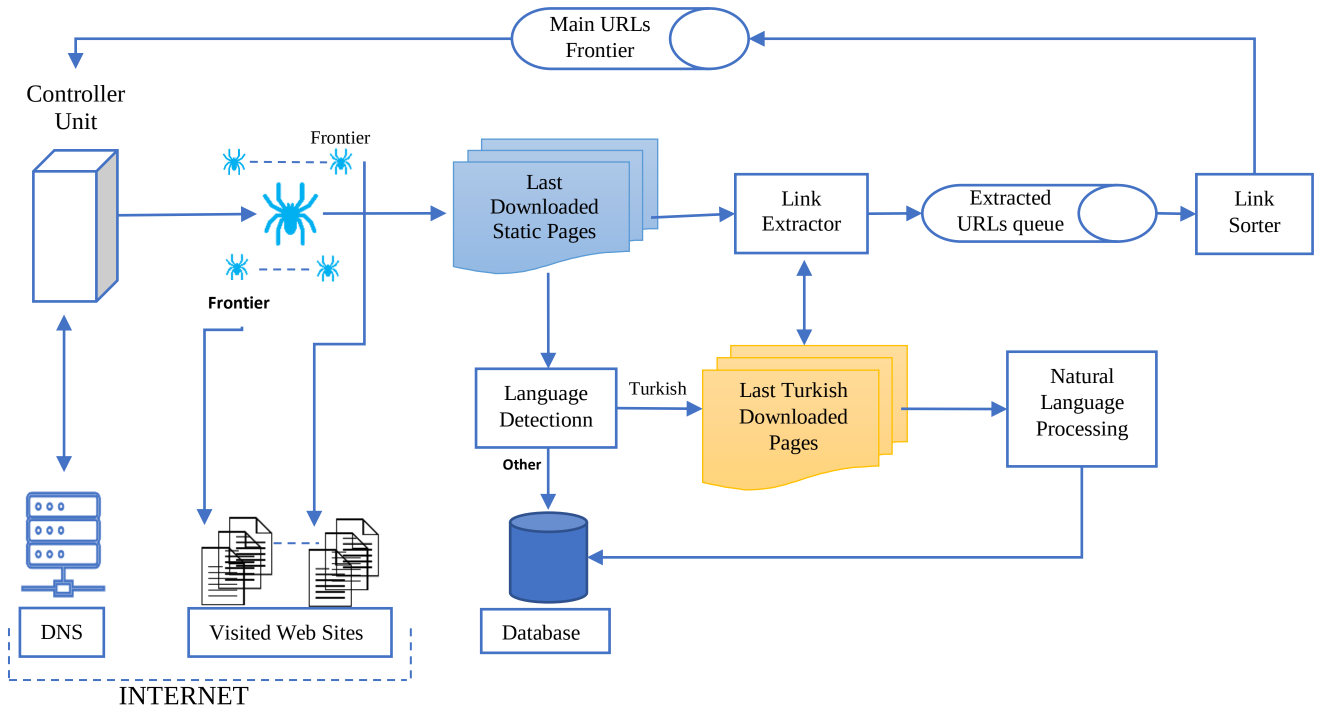

Figure 1. The structure of the developed focused crawler.

- *Link Sorter Unit*: this unit is responsible for getting the URLs from the "ExtractedURLsQueue." Those URLs are added to the appropriate position in the main URL frontier(s) based on the sorting process. In the work of [6], the backlink count algorithm has been adopted to sort the URLs. In general, a website's backlink rank is determined by the number and quality of websites that refer (contain a link) to that website. In this study, other essential factors, such as the originality score of the website content, the page keywords, and their TF-IDF, were used in ranking the URLs to improve the ranking process further.
- *Natural Language Processing Unit (Turkish Language Processing unit)*: this unit is responsible for classifying the downloaded Turkish web pages. It uses the sentence level for language processing. The sentence level analyzes each sentence individually to classify the web pages. This level adds more flexibility than the other levels; for instance, it can differentiate each sentence type, such as the objective, subjective, etc.,

In more detail, the classification process is done based on the page's contents and metadata, and its main steps are:

1. Preprocessing: data quality is a critical issue in data mining systems and classification in particular. Web pages should be preprocessed to remove unwanted and useless parts. Otherwise, such parts may lead to an inaccurate classification process. The preprocessing consisted of Normalization, Stemming, and Lemmatization in this work. In more

detail, as shown in Table 1, Normalization contains some sub-steps, such as:

1. Case folding, i.e., converting the whole text into the same state (upper or lower).
2. Segmentation refers to dividing large parts of a text (document) into pieces larger than words (e.g., paragraphs or sentences).
3. Removing useless parts, tags, punctuations, and stop words. In other words, in this step, we extracted the content of the page body, such as text, in addition to the page's metadata, such as caption, and then eliminated the HTML tags "<p>, <a>, <table>, etc.,”. Also, we delete the punctuation and stop words. The aim here is to remove parts that do not carry any direct information, such as conjunction, and preposition, for example, “the” and "for” in English, “böyle” and “daha” words in Turkish which do not affect the content in any sense. In addition, excluding widespread words and thus having no discriminating effect is also used as a filtering technique.
4. Indexing numbers by converting them to equivalent words.
5. Handling typos and misspellings (spatially, the ones done intentionally), i.e., multiple letters, can be repeated while writing some words. For example, “seviooorummm” (I love you) and “çooook” (Many) are captured, and the repetitive letters are reduced to one. In addition, Turkish Deasciification can be considered one of the typos correction steps. There are eight different special characters (“ç,” “¸s,” “g,” “ı,” ˘ “ö,” “ü”) in Turkish that are not in English. Since some smart devices do not have these Turkish characters, and even for simplicity, the majority of users are using the closest ASCII characters (“c,” “s,” “g,” “i,” “o,” “u”) in informal correspondence, especially on platforms such as chat, forum, social media, and SMS. For example, the proposed unit will convert the word “kizmis” written in ASCII characters into the Turkish form “kızmış.”

Table 1. Sample of the normalization input and its corresponding output.

Normalization	Input Text	Output Text	English Meaning
Slang word	slm	selam	Hello
Repeated characters	çoooook	çok	Many
Emo style writing	$arkı4U	Şarkı senin için	Song for you
Capitalization	GaziAntep	gaziantep	Name of city
Deasciification	hoslanmmadim	hoşlanmadım	I did not like it

- *Stemming*: briefly, it works on eliminating all kinds of affixes to obtain the stem (root) for each word, where for example, the English word eating is converted to eat, and the words trouble, troubling and troubled are converted to troubl; similarly the Turkish words “arama” (calling) and “araştırıcı” (searching) are derived from the same word “ara.”
- *Lemmatization*: it works on capturing canonical forms of the word, i.e., it is the process of converting all the inflected forms of a word into the identified word's lemma. For example, the word operating is converted into operate.

2. Feature Extraction: to classify any text, it must be converted into numerical values. This process can be done using the Vector Space Model [12, 25], an arithmetic model mainly used to represent the text by weights and vectors. Term Frequency-Inverse Document (TF-IDF) is one of the old fashion, frequently used methods. While, recently word embeddings, which can efficiently preserve contextual similarity while representing the words in low dimensional vector space and producing a similar representation for similar meaning words, were able to achieve outstanding performance. Bidirectional Encoder Representations from Transformer (BERT), Embeddings from Language Model (ELMO), and Generalized Autoregressive Pretraining for Language Understanding (XLNet) are state-of-the-art embedding approaches. In this paper, we have investigated the performance of the mentioned approaches to find the most suitable one for the Turkish language. The detail of these embeddings approaches is summarized below.

   a) *BERT*: this algorithm started to be used in 2018 by Google in NLP for multiple purposes, such as better understanding the queries and providing more accurate results by evaluating the words in the queries. In general, BERT is an NLP-based two-dimensional embedding model. It is a trainable model that can handle large plaintext corpus [12].
   b) *ELMO*: it uses two bidirectional language model layers, i.e., forward transition and backward transition. These two layers consider the context of the previous and the following words for the current input word, which creates intermediate vectors in forwarding and backward transition that will be fed into the next layer. The weighted sum of the raw word vectors and two intermediate word vectors is produced as the final representation. Hence, ELMO can assign different vectors for the same word based on the context of the sentence. Elmo provides an advantage over traditional embedding algorithms such as Word2vec and Glove, which have a single numerical representation regardless of the meaning of words and/or their location in the text [21].
   c) *XLNet*: it is one of the newest NLP models and is based on recursive transformer architecture. Also, XLNet is a language model that gives the common persistence of a set of words. Unlike the

other models, it calculates the word's vector based on permutations of all sentence words. While doing this process, XLNet avoids the assumption of independence and fine-tuning [25].

3. Classification: this paper proposed a new efficient CNN model that can classify Turkish webpages in general and the web pages downloaded by the proposed crawler in particular. As shown in Figure 2, the developed CNN model consists of multiple layers such as embedding, global max pool, and dense. The details about the main layers of this model are summarized below.

- *Embedding layer*: this layer was originally developed to improve the performance of CNN when processing natural languages (text). In this work, it is the first used layer. In general, the input of this layer is the features of the current web page, i.e., F_D, which was obtained in the previous step, and its size [N, 150], where N is the number of the samples (downloaded webpages). This layer consists of 300 neurons, whose weights are multiplied by the input to produce an output, i.e., Embedding output that we refer to by E_D, and it is of size [N, 150, 300].
- *Pooling Layer*: it is the second used layer, and it is used for reducing the size of the activation maps (down-sampling the feature maps). Given that the input of this layer is the E_D, this layer works on reducing the size to [N, 300], which we call E_{maxD}.
- *Dropout*: this layer works by ignoring some randomly chosen neurons and their connections (incoming and outgoing) during the training phase. Hence, it temporarily drops out some neurons for every training sample. The main aim of this process is to prevent the model from overfitting. In the proposed CNN model, a 0.2 Dropout is allocated after the second layer, i.e., global max1D, and another 0.2 dropout is allocated between the last two layers, i.e., the fully connected layers.
- *Fully Connected Layer (dense layer)*: its name came from the fact that all the neurons in the layer are connected to those in the next layer. In this work, two dense layers are used, where the input of the first one is of size [N, 300] and the output of size [N, 2* C_n]. Finally, the second dense layer produces a membership for each sample to each of the C_n classes. Based on the used datasets in this study, C_n is equal to 14.

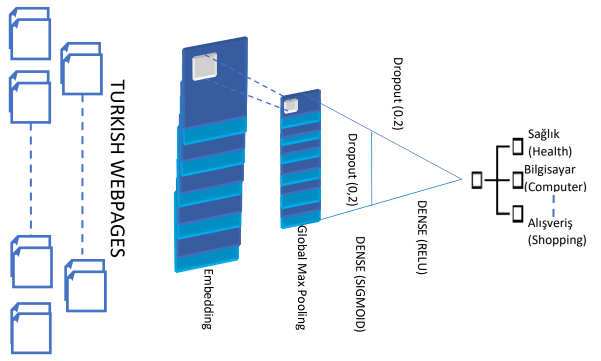

Figure 2. A visualization of the developed CNN architecture. The network consists of an embedding layer, a pooling layer followed by two dense layers.

It is worth mentioning that processing dynamic webpages and supporting Asynchronous JavaScript And XML (AJAX) is kept as future work. This is due to the fact that, as shown in [6], supporting AJAX is more complicated than processing static web pages, and this process occupies large portions of the server(s) resources such as memory, CPU cycles, etc.

4. Experiments

In this section, the performance of the proposed system for supporting the Turkish language was investigated through multiple experiments. Multiple databases have been used to ensure the robustness of the developed system, whose details are shown in the following subsection.

4.1. Datasets, Experiments Setups, and Evaluation

In this work, two datasets that are publicly available [8, 13] are used for training the developed CNN model. In contrast, the third one, which contains 50,000 Turkish web pages downloaded by the developed crawler, is used to test the proposed model. Related to the datasets of [8, 13], the first one contains "28,567" Turkish web pages that were manually annotated into 13 main categories such as Alışveriş (Shopping), Bilgisayar (computer), Sağlık (Health), etc., The second one, consisting of "22,431" is also manually annotated into 14 main categories.

Related to implementing the proposed CNN model, Python and libraries such as Keras, Pandas, and NumPy are used. To support the Turkish language, we used the library Turkish-deasciifier presented in [3] for the deasciification process and Turkishnlp for text normalization. For stemming, the python implementation of the TurkishStemmer presented in [3] is used. In addition, the preprocessing and feature extraction functions were imported from both Keras and Sklearn. Related to BERT, we used

1. The BERT tokenizer imported from the library "bert-for-tf2" (the BERT library for TensorFlow).
2. The "bert base multilingual uncased," which

supports 102 languages, including the Turkish language.

About Elmo, we used "tensorflow_hub" and the third version of its model, i.e., elmo3. Finally, XLNetConfig, XLNetForSequenceClassification, XLNetTokenizer, and XLNetModel were imported from the "transformers" library to use and retrain the XLNet. The three mentioned embedding systems were trained using:

1. The "Turkish Wikipedia dump" dataset.
2. The Turkish pre-trained word vector was initially trained using the "Turkish CoNLL17 corpus," consisting of 3633786 vocabulary size. We decided to use the last one as it achieved the best performance based on our preliminary experimental investigation.

Also, the accuracy, precision, recall, and F1 score, which are standard metrics for evaluating classification systems, are used in this work. It should be noted here that two of the performed experiments were not added due to space constraints. The first one investigated the effect of preprocessing webpages vs. using the raw data, and its results showed that preprocessing can boost performance in the range of 7%-15%. The second one aims to observe the daily number of web pages downloaded by the developed crawler using a different number of threads (between12-128). On average, our crawler could download 300 thousand pages daily, using two PCs with Intel(R) Core (TM) i7-9750H CPU @ 2.60GHz and 32.0 GB.

- **Experiment #1: Comparing the performance of feature extraction techniques**

One of the main steps in text classification is to represent the text numerically; currently, embedding approaches such as BERT, Elmo, and XLNet are considered state-of-the-art and the best option. In this experiment, the performance of the mentioned embeddings was investigated to find the most suitable one for the Turkish language. In more detail, each of these embeddings was integrated with the developed CNN model, and their performance was investigated. Also, to ensure the scalability and robustness of the proposed system, this experiment was repeated five times (5 iterations). As shown in Figure 3, with the improvement achieved by embedding feature extraction techniques in general, all studied approaches have a pretty good performance. However, BERT was able to outperform the others. Hence, BERT is the most appropriate for building an efficient Turkish language classification system.

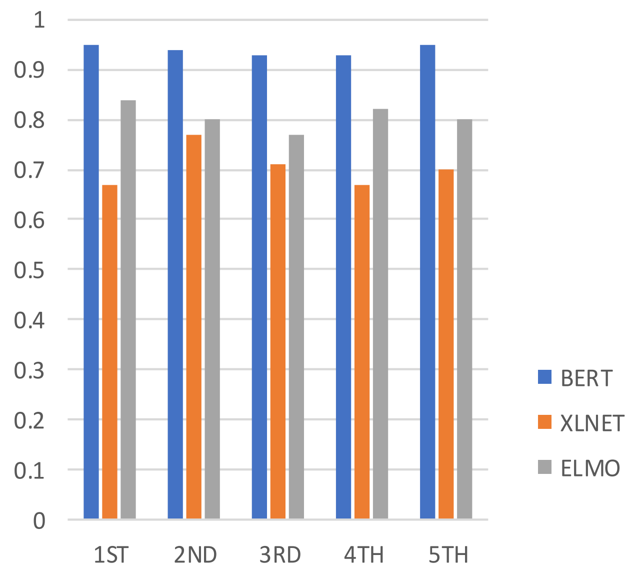

Figure 3. The accuracy of BERT, Elmo, and XLNet techniques.

- **Experiment #2: Comparing our approach with the state-of-the-art CNN based text classifiers**

As mentioned before, an efficient classifier is essential for webpage classification in general and topic-specific crawling in particular. To validate the proposed classifier, we compare its performance against seven state-of-the-art CNN models, i.e., "TurkishCNN" [11], "TurkishRand" [16], "Onelayer" [14], "VdCNN" [10], "SensitivityAnalysis" [28], "CharCNN" [27], and "CharTCN" [7].

In detail, we have re-implemented the aforementioned approaches and used their hyperparameters as stated in their corresponding references. As shown in Figure 4, it is clear that the developed approach has significantly outperformed all approaches when processing Turkish webpages, where the accuracy improvement is around 47% as compared to four of the studied models, i.e., "TurkishCNN," "CharCNN," "CharTCN," and "Onelayer" models, whereas the lowest improvement is 10% for both "VdCNN" and "TurkishRand" models.

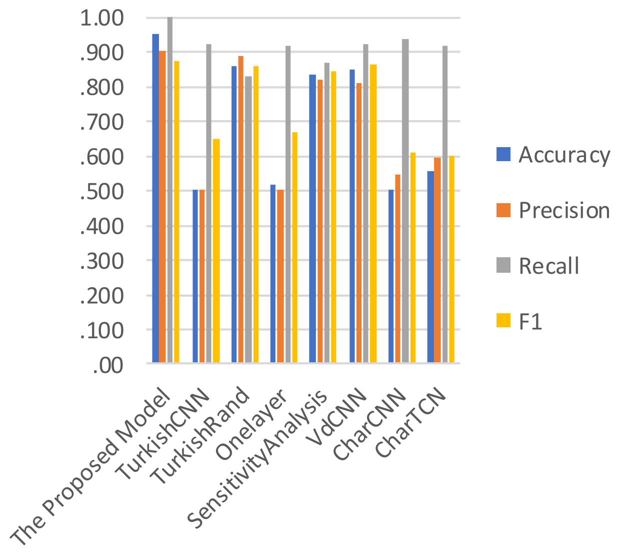

Figure 4. The accuracy, precision, recall, and F1 for all studied models.

5. Conclusions and Future Works

A focused crawler that employs parallel principle and CNN learning-based to efficiently supports the Turkish language is presented in this paper. The developed crawler consisted of multiple units that could work in parallel to help process the massive amount of published websites. Also, the proposed natural language processing unit works on categorizing Turkish web pages efficiently. The experimental works have illustrated the superior performance of the developed model compared with seven state-of-the-art CNN classification systems. Also, BERT was found to be the most efficient embedding when processing the Turkish language.

As a feature work, the proposed crawler will be documented in detail and published as an open source. Such a crawler can be used to collect and classify web pages and build some valuable datasets. Another direction for future work can be building a second version of the proposed crawler that can process dynamic webpages and AJAX.

References

[1] Abd S., Alsajri M., and Ibraheem H., "Rao-SVM Machine Learning Algorithm for Intrusion Detection System," *Iraqi Journal for Computer Science and Mathematics*, vol. 1, no. 1, pp. 23-27, 2020.

[2] Aggarwal K., "An Efficient Focused Web Crawling Approach," *in Proceedings of the Software Engineering*, Singapore, pp. 131-138, 2019.

[3] Akın A. and Akın M., "Zemberek, An Open Source NLP Framework for Turkic Languages," *Structure*, vol. 10, no. 2007, pp. 1-5, 2007.

[4] Ali A., Hussain Z., and Abd S., "Big Data Classification Efficiency Based on Linear Discriminant Analysis," *Iraqi Journal for Computer Science and Mathematics*, vol. 1, no. 1, pp. 9-14, 2020.

[5] Alqaraleh S., "Novel Turkish Sentiment Analysis System Using ConvNet," *The International Arab Journal of Information Technology*, vol. 18, no. 4, pp. 554-561, 2021.

[6] Alqaraleh S., Ramadan O., and Salamah M., "Efficient Watcher Based Web Crawler Design," *Aslib Journal of Information Management*, vol. 67, no. 6, pp. 663-686, 2015.

[7] Bai S., Kolter J., and Koltun V., "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arXiv, preprint arXiv:1803.01271*, 2018.

[8] Biricik G., Diri B., and Sönmez A., "Impact of a New Attribute Extraction Algorithm on Web Page Classification," *in Proceedings of the International Conference on Data Mining*, Las Vegas, pp. 481-485, 2009.

[9] Ciftci B. and Apaydin M., "A Deep Learning Approach To Sentiment Analysis in Turkish," *in Proceedings of the International Conference on Artificial Intelligence and Data Processing*, Malatya, pp. 1-5, 2018.

[10] Conneau A., Schwenk H., Barrault L., and Lecun Y., "Very Deep Convolutional Networks for Text Classification," *arXiv, preprint arXiv:1606.01781*, 2016.

[11] Demirci G., Keskin Ş., and Doğan G., "Sentiment Analysis in Turkish with Deep Learning," *in Proceedings of the IEEE International Conference on Big Data (Big Data)*, Los Angeles, pp. 2215-2221, 2019.

[12] Devlin J., Chang M., Lee K., and Toutanova K., "Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding," *arXiv, preprint arXiv:1810.04805*, 2018.

[13] Hüsem S. and Gülcü A., "Categorizing the Turkish Web Pages by Data Mining Techniques," *in Proceedings of the International Conference on Computer Science and Engineering*, Antalya, pp. 255-260, 2017.

[14] Kim Y., "Convolutional Neural Networks for Sentence Classification," *Arxiv Preprint, Arxiv:1408.5882*, 2014.

[15] Kumar M, Bindal A, Gautam R, Bhatia R., "Keyword Query Based Focused Web Crawler," *Procedia Computer Science*, vol. 125, no. 18, pp. 584-590, 2018.

[16] Kurt F., "Investigating the Performance of Segmentation Methods with Deep Learning Models for Sentiment Analysis on Turkish Informal Texts," Master Theses, Middle East Technical University, 2018.

[17] Lee J., Bae D., Kim S., Kim J., and Yi M., "An Effective Approach to Enhancing a Focused

Crawler Using Google," *The Journal of Supercomputing*, pp. 1-18, 2019.

[18] Li H., Zhang Z., and Xu Y., "Web Page Classification Method Based on Semantics and Structure," *in Proceedings of the 2nd International Conference on Artificial Intelligence and Big Data*, Chengdu, pp. 238-243, 2019.

[19] Liu D., Lee J., Wang W., and Wang Y., "Malicious Websites Detection via CNN based Screenshot Recognition," *in Proceedings of the International Conference on Intelligent Computing and Its Emerging Applications*, Tainan, pp. 115-119, 2019.

[20] Pant G., Srinivasan P., and Menczer F., "Crawling the web," *Web Dynamics: Adapting to Change in Content, Size, Topology and Use*, vol. 153, pp. 153-177, 2004.

[21] Peters M., Neumann M., Iyyer M., Gardner M., Clark C., Lee K., and Zettlemoyer L., "Deep Contextualized Word Representations," *in Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, pp. 2227-2237, 2018.

[22] Sekhar S., Siddesh G., Manvi S., and Srinivasa K., "Optimized Focused Web Crawler with Natural Language Processing Based Relevance Measure in Bioinformatics Web Sources," *Cybernetics and Information Technologies*, vol. 19, no. 2, pp. 146-158, 2019.

[23] Wang C. and Chen Y., "Topical Classification of Domain Names Based on Subword Embeddings," *Electronic Commerce Research and Applications*, vol. 40, pp. 100961, 2020.

[24] Yadav D., Sharma A., and Gupta J., "Users Search Trends on WWW and Their Analysis," *in Proceedings of the 1st International Conference on Intelligent Interactive Technologies and Multimedia*, Allahabad, pp. 59-66, 2010.

[25] Yang Z., Dai Z., Yang Y., Carbonell J., Salakhutdinov R., and Le Q., "Xlnet: Generalized Autoregressive Pretraining for Language Understanding," *Advances in Neural Information Processing Systems*, vol. 32, pp. 5753-5763, 2019.

[26] Yildirim S., *Deep Learning-Based Approaches for Sentiment Analysis*, Springer, 2020.

[27] Zhang X., Zhao J., and LeCun Y., "Character-Level Convolutional Networks for Text Classification," *Advances in Neural Information Processing Systems*, vol. 28, pp. 649-657, 2015.

[28] Zhang Y. and Wallace B., "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification," *arXiv, preprint arXiv:1510.03820*, 2015.

[29] Zhao Q., Yang W., and Hua R., "Design and Research of Composite Web Page Classification Network Based on Deep Learning," *in Proceedings of the IEEE 31st International Conference on Tools with Artificial Intelligence*, Portland, pp. 1531-1535, 2019.

Saed Alqaraleh is an Assistant Professor at Hasan Kalyoncu University, Gaziantep, Turkey. He completed his Ph.D. in Computer Engineering (2015) from Eastern Mediterranean University, Cyprus. His present research areas include Information retrieval and security, Natural Language Processing, Visual Place Recognition, and Machine learning.

Hatice Meltem Nergiz Sirin is a Research Assistant at Hasan Kalyoncu University, Gaziantep, Turkey. She is a first-year Ph.D. student in Computer Engineering program at Hacettepe University, Ankara. Her present research areas include Natural Language Processing, Computer Vision, and Robotics.