Journal of Theoretical and Applied Information Technology
15th December 2019. Vol.97. No 23

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CLASSIFYING ARABIC TEXT USING DEEP LEARNING

MOHAMED GALAL, MAGDA M. MADBOULY, ADEL EL-ZOGHBY
Information Technology Department
Institute of Graduate Studies and Research - Alexandria University
163 Horreya Avenue, Shatby-P. O. Box: 832, Alexandria, Egypt
E-mail: galaldev@gmail.com, mmadbouly@alexu.edu.eg, zoghabi@gmail.com

ABSTRACT

Nowadays, the volume of data offered on the Internet is growing every moment, and the necessity to analyze these data and convert to useful information increased. There are several types of research exploring techniques to deal with Text Classification (TC) in many languages; however, In Arabic, the researches are limited. TC is the process of categorizing text document into classes or categories according to the text contents. This research will focus on classifying Arabic Text using a Convolution neural network (CNN), which considered one of deep learning (DL) methods, as it achieved an excellent result in different Natural language processing (NLP) project types [1],[2],[3]. We also introduced a novel algorithm to group similar Arabic words based on extra Arabic letters and word embeddings distances. We named this algorithm as GStem.

Keywords: *Arabic Text Classification, Gstem, Neural Network, Deep Learning*

1. INTRODUCTION

TC is considered one of the most important tasks in extract information especially with the fast growth of text data every moment. It aims to allocate text in test data to its predefined class based on its text content [4]. TC is quite a challenging field in text mining as it requires reprocessing steps to convert unstructured data to structured information [5]. Most researches of TC aimed to classify documents written in English or other languages such as Spanish, French, German, Chinese and Japanese using approaches such as Support Vector Machine (SVM), decision trees, K-nearest neighbors (KNN), Naïve Bayes (NB), and Artificial Neural Network (ANN). However, in Arabic, there is little ongoing research in automatic Arabic text classification (ARTC)[6].

1.1 Arabic Language Complexity

Arabic Language is different in structure and grammar form any other language. So, it is difficult to reuse the research which was constructed for English with those which use Arabic [7]. One of the most important problems of ARTC is that the meaning of the text in Arabic depends heavily on the surrounding context. Another problem is morphological analysis because all words in Arabic are derived from a list of roots with 3 up to 6 constants which means that the language is derivational[8]. Also, Words in Arabic can be masculine or feminine likewise words are formed by adding affixes to word root. Definite articles ال (The), conjunctions, particles, and other prefixes can be affixed to the beginning of a word, whereas suffixes can be added to the end [9]. All the previous characteristics made the classification of text written in Arabic is a difficult task.

1.2 Text Classification Techniques

In order to deal with text data, we need some preprocessing to select the features which semantically represent the document and remove other features that is not. This process which extracts the important features that represent the training dataset is named feature extraction (FE)[10].

Machine learning (ML) considered one of the important techniques used in TC . First, text represented as feature vectors Then the chosen learning classifier (such as SVM, NB, ANN or another classifier algorithm) trained on the feature vectors to produce an ML model responsible for predicting text documents classes [11]. Deep Learning (DL) is an extended field of ML deal with different types of data (such as Images, speech, Text, videos …. Etc) and has a collection of algorithms useful to learn supervised and unsupervised fashion.These techniques have achieved good results in numerous fields including Natural Language Processing (NLP) and ARTC[12].

The Convolutional Neural Network (CNN) is one of the important DL techniques originally deals with images data (2D data structure). The CNN is an

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ANN that can process input data through convolution layers where each computation unit responds to a small region of input data. Recently Some researchers apply CNN to text date by converting input documents to 2D data structure so that each unit in the convolution layer responds to a small region of a text document which is one sentence only [13]. CNN produced outstanding results in different NLP applications such as search query retrieval, semantic parsing, sentence modeling, or other NLP projects [1]. In this work, CNN as a classifier was used for classifying text data in Arabic language.

2. LITERATURE SURVEY

The ARTC was studied in [14], to evaluate the performance on Arabic documents using NB algorithm which one of the simplest algorithms applied to TC in English. The author aimed to gain some insight as to whether Arabic document categorization using NB is sensitive to the root extraction algorithm used or to different data sets. He collected another dataset from the web contains 300 web documents for each of five categories. The author started his work by text preprocessing by removing stop words , stripe the vowels from the full text and use of a root extraction process, he stated that the aim of root extraction process is to transform of all Arabic word derivatives to their single common root or canonical form which is very useful in terms of decreasing the indexing structure with taking advantage of the semantic relationships between the different forms of the same root. He used the Arabic root extraction technique and It compared to other stemming or root extraction algorithms, then he stated that the performance of this algorithm results over 97% for discovering the right root in documents. Then the author implements classification using NB after splitting the dataset to training and testing datasets. After that he conducted several experiments first, he performed cross-validation using all the words in the documents, second cross-validation experiments based on feature selection (using a subset of terms/roots only), third experiments based on an independently constructed evaluation set (manually selected documents that best represent the variability in the category). In the end, the author concludes that the performance of the NB algorithm in classifying Arabic documents is not sensitive to the Arabic root extraction algorithm.

A new approach of stemming the Arabic word was studied in [15]. First the author mentioned that the problem of stemming on Arabic language is different from English because the Arabic word can contain prefixes, suffixes, and infixes such as word "المعلومات" which means "information", consists of the following parts: "ال" represent prefix-part, "ات" represent infix part, "و" represent the suffix-part and "علم" which represent the root part. To solve the problem of stemming Arabic words the author proposed a new model based on a backpropagation ANN and the Arabic language extra letters. The author stated that Arabic linguists have identified the commonly used letters in these affixes and presented them in the following set (Arabic extra letters): {س,أ,ل,ت,م,و,ن,ي,ه,ا} Then the Arabic linguistics experts classified these letters according to their frequency in appearing as affix letters, The highest frequently used are {ا,و,ي}, then {أ,م,ن,ت} and the lowest are {س,ه,ل}. After that the author started his model by represent each word by its letters according to which group of the previous three groups for example if the letter is any of {ا,و,ي} then it will present by 1, if in {أ,م,ن,ت} it will present by 2, if in {س,ه,ل} it will present by 3 and if another letter then it will present by 4. The output layer is a vector with 1 value for each input letter represent the word root otherwise will be 0. The author trained his model in training set with 9000 samples then he tested it with 250 samples and gives an accuracy of about 84%. In our research, we tried to use the linguistic rule of Arabic extra letters to group similar words by a new proposed algorithm called GStem.

In this research [16] The author proposed an improved KNN Classifier for using in ARTC depending on two types of indexing Word- Level N-Grams and comparing with Single Terms. After the processing of all text data, each text document represented by all of its word-level unigrams and bigrams along with the frequency of each in this text document. The author used a Document Frequency Features Selection to reduce the dimensions of the features, After the weight of each N-gram is calculated using TF-IDF, KNN classifier is used as a TC method. The same steps are applied to classifying all documents based on the bag of words (BOW) approach to compare each approach accuracy. His dataset collected from the web and contains 1445 documents that vary in length and classified into nine categories, and divided this dataset by 60% for the training set and the rest for testing. He starting his work by performing Arabic text preprocessing such as (Removing non-Arabic letters, digits, single Arabic letters, removing stopwords, … etc.) and he mentioned that text preprocessing is very useful because it reduces the index size, increases the accuracy and unifies the classification activities. Then he represented each

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

document by a vector of its unigrams and bigrams weights, in addition to that, every document represented by a vector of its single terms weights for comparison purpose. He used document frequency feature selection for dimensionality reduction and all features that have document frequency value above a predefined threshold (which is 3 in his experiment) are selected along with feature frequency and Weights are calculated using TF-ID. Then the represented weights document passed to KNN classifier to perform the classification. Finally, he tested his model with different document indexing approaches and concluded that the use of N-gram to represent each document produces better performance than using single terms. The average accuracy in the case of using N-Gram is 73.5% while with Single terms indexing, it is 66.8%.

A comparison between SVM and NB and their effect on ARTC was introduced in [4]. first, the author stated the higher complexity of the Arabic language than the English language in the TC problem because Arabic language is highly inflectional and derivational language and this increases the complexity of monophonic analysis. Then the author used an Arabic dataset collected from Saudi Newspapers, contains about 5121 Arabic article different lengths that belongs to seven classes. Then he described the technique he used by first do some text preprocessing such as (removing digits and punctuation marks, normalizing some of Arabic letters, filtering all none Arabic letters and removing stopwords). After that, the author starts to apply SVM and Naïve Byes classifier to the preprocessed text data to evaluate the result. Then he used three evaluation measures (Recall, Precision, and F1). In the end, he concludes that the three measures average obtained against SNP Arabic data sets indicated that the SVM outperformed NB regards to F1, Recall and Precision measures.

A study of the impact of preprocessing methods and classification algorithms on the accuracy of Arabic document categorization was introduced in [10]. The author gives an overview of the Arabic language structure and complexity compared to English such as

- The Arabic language has richer morphology than English because of in Arabic many words with different meaning can be generated by one root
- Arabic has prefixes, suffixes, and infixes but English has only prefixes and suffixes
- In Arabic, some words can have the same letters but different meanings according to their location in the context. In other hands may one word have more than one lexical category.
- Proper names, acronyms, and abbreviations start with capital letter in English, however, in Arabic there is no upper or lower case for letters.
- There is no free benchmarking dataset for ARTC, however, in English there are more.

After that, the author described a framework for Arabic document categorization starting with text data preprocessing such as Word Normalization, Text Tokenization, Stop Word Removal and Stemming using light stemmer [17]. After that he used a TF-IDF method for text representation in vector space then he used chi-square testing for feature selection. In the classification step, the author compared KNN, Naïve Byes and SVM and then he noted that there is no free benchmarking dataset for Arabic document categorization so he collected his own dataset from several published papers for TC contains 32,620 documents divided into 10 categories that vary in length and number of documents. After that F1 measure selected for evaluation. At the end the author concludes that the normalization task helped to improve the performance and provided a slight improvement in the classification accuracy regardless of feature size and classification algorithms, removing stopwords has negative impact when combined with other preprocessing tasks in most cases and stemming techniques improve the classification accuracy insignificant way if applied alone or combined with other preprocessing methods. The results also showed that SVM is better than NB and KNN in all experiments.

The using of CNN for solving the TC problem was introduced in [1]. At first, the author stated that DL models have achieved remarkable results in computer vision and speech recognition on the other hand in natural language processing many DL works concerning learning word vector representations by the neural language model. Then he stated that CNN which is originally invented for computer vision have achieved excellent results in NLP tasks such as sentence modeling, search query, semantic parsing, and other NLP applications. After that, he introduced his proposed model which is a simple CNN on top of word vectors obtained from an unsupervised neural language model. These vectors were trained by [18] on 100 billion words of Google News, and are publicly available. The first step on the model is text data represented as word vectors then all input text sentences are padded to be in the same length, this passed to convolution filters to produce new features and these filters are applied to each window

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of words in the sentence to produce a features map. After that, a max-over- time pooling operation is applied to capture the most important features. After that the learned featured passed to a fully connected softmax layer whose output is the probability distribution over labels. To prevent co-adaptation of hidden a dropout employed with a constraint on l2-norms of the weight vectors. The author used in his experiments several datasets such as Movie reviews, Stanford Sentiment Treebank, Subjectivity dataset, TREC question dataset, Customer reviews of various products and MPQA dataset each dataset is divided to train and dev set but if the dataset has no standard dev set then he split it by 10% for dev set. The training is done through stochastic gradient descent over shuffled mini-batches with the Adadelta update rule and hyperparameters such filter windows of 3, 4 and 5 every filter has 100 feature maps, a dropout rate of 0.5, l2 constraint of 3 and mini-batch size of 50 are used. He built his experiments based on several variations such as CNN-Rand which randomly initialize word vectors, CNN-Static which used pre-trained word vectors, CNN-non-static which pre-trained word vectors but fine-tuned for each task and CNN-multichannel which a model with two sets of word vectors. The results show that (CNN-rand) does not perform well but a simple model with static vectors (CNN-static) performs remarkably well in comparison to more sophisticated DL models that utilize complex pooling schemes or require parse trees to be computed beforehand. The results also show that fine-tuning the pre-trained vectors for each task gives still further improvements (CNN-non-static). He finally concludes that a simple CNN with one layer of convolution performs remarkably well and the unsupervised pre-training of word vectors is very important in DL for NLP.

In [19], the author tried to solve the problems of word embeddings (WE) and TC for the Arabic language. He started by describing the difficulty of Arabic language in sentiment classification because it has a very complex morphology and structure, monophonically analysis on Arabic is more difficult because of inflectional and derivational nature of the Arabic language and dialectal Arabic is used more than modern standard Arabic on the internet posts and topics. To solve the problem of WE and TC for the Arabic language the author crawled the web to build billions of Arabic words corpus then created a WE model to produce Arabic word representations using the crawled corpus finally he trained a CNN on top of WE to perform TC. In the crawling phase, the author used an open source free crawler called then he created a huge web-crawled corpus for modern standard Arabic and dialectal Arabic text. As a part of text preprocessing the author used a technique to extract and my quality phrases from a large text. After that, the author used Word2Vec to train WE. In the training phase, the author uses a CNN architecture similar to [1]. To evaluate the model the author runs his experiments on different corpora from two different domains (reviews and tweets). Finally, he concluded that a good performance obtained by using high quality of the data, high dimensionality vectors have good performance with a large corpus using a word embedding which pre-trained before.

The using of Word and Document Embeddings as a representational basis rather than depending on text preprocessing and BOW representation was studied in [20]. For WE the author used a Continuous bag of words and the Skip-Gram models of the word2vec [18] and the Glove model [21] then he used the document vector model [22] document level embedding. The comparison is done between document embeddings techniques against BOW representation using word count and TFIDF weighting scheme. When BOW representation used the author used many text preprocessing techniques such as stop word removal, normalization, and stemming by (Khoja stemmer [23], Light 10 stemmer [17], Moataz stemmer [24], and Tashaphyne stemmer1). The author collected a large amount of raw Arabic texts containing about 216 million words due to train the word and the document embedding models. The classification process is done using SVM algorithm on the OSAC data set [25] then the results are evaluated using the average weighted precision, recall, and F1 measures. In the end, the results show that document embeddings representations outperform all BOW representations which obtained using text normalization and different stemming algorithms using all of word count and TFIDF.

A study investigating various DL techniques such as CNNs and Long Short-Term Memory (LSTM) recurrent neural networks for sentiment analysis of Arabic language published here [26]. The author compared different models with multiple techniques. He finally concludes that using word2vec vectors updated during learning achieves the highest results in nearly all cases and proposed combined LSTM outperforms all other models.

The impact of stemming algorithms as FE for ARTC was studied in [27]. The author compared two stemming techniques the stemming [23], which finds the three-letter roots for Arabic words and light stemming which only drops the prefixes and suffixes

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

from the Arabic words then used the KNN as a classification algorithm. The author concludes that Light stemming gives better results than the stemming approach because the stemming usually affects the meanings of the words.

As mentioned in previous researches the Arabic language has little ongoing research in automatic ARTC in comparison to English and most of these researches using ML techniques is relying on traditional classifiers (such as SVM, KNN, NB, and Regular ANN). However, DL is recently introduced and perform well in the TC process. Some researches perform a preprocessing step but depending on a predefined dictionary to perform stemming or perform light stemming by only drops the prefixes and suffixes from the Arabic. So our contribution in this paper is to introduce a new technique can detect similar Arabic words which share the same root without depending on any dictionary. We employ Word2Vec technique which has the ability to represent near words in meaning to near vectors and use the Arabic Extra Letters rule (Grammar rule which says that, all root words can be converted to another word by adding one or more letter from {س,أ,ل,ت,م,و,ن,ي,ه,ا} to the original word) with minimum edit distance (method to compute distance between two words based on its letters). We implemented this technique and introduced a new algorithm called G-Stem achieving this task. After that, we employed a deep CNN to perform the task of TC to test the effect of using DL in our ARTC task.

3. PROPOSED SYSTEM

Our goal is to build a model used to classify different text documents to its predefined classes. We crawled the web and collected an Arabic news dataset formed from four categories (art, economic, accident, and politics) each category has 1500 sample and the total samples of the dataset are 6000. We also published our dataset for future use in other researches [1]. We build our model starting with text preprocessing to remove any unnecessary data such as digits, punctuation marks, duplicate spaces, and none Arabic words and prepare text data for feature selecting. Then we convert all text data to vector space using the Word2Vec technique because the type of representation that the Word2Vec does gives the words has the same meaning nearby representation which gives a huge positive impact to the classification process [28]. After that, we introduced and implement a new technique to group Arabic words that share the same root based on Arabic words extra letters and word vectors distances. After that, we trained a CNN which is a slight variant of architecture for classifying Text data published in [1] and use it to classify Arabic text documents to its predefined classes. Finally, we calculated the Accuracy of the model and analyze the result of the model architecture showed in Figure 1.

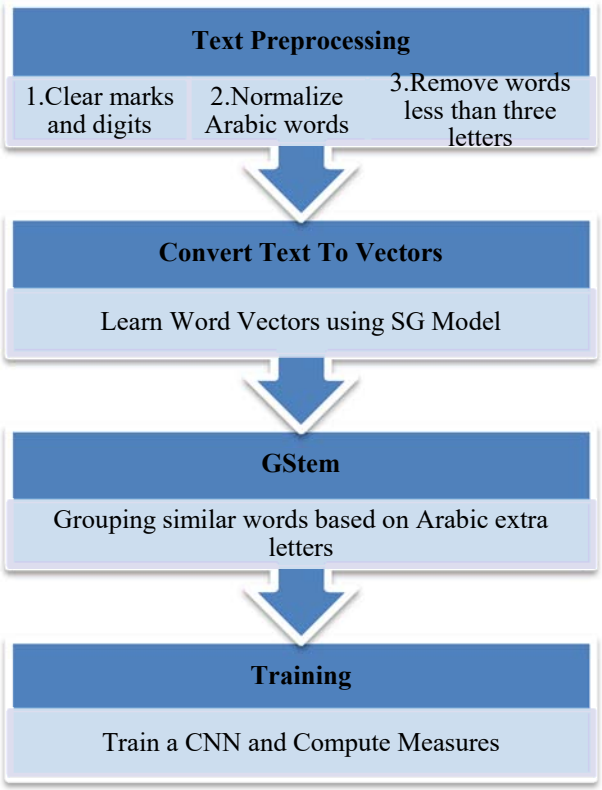

Figure 1 Model Architecture for ARTC

3.1 Text Preprocessing

Input text requires some preprocessing before it can be used in classification so the first step after selecting the dataset is to clean input text data and remove any noise from it. This can be achieved by the following. The first step the system will clear digits, punctuation marks, duplicate spaces, and none Arabic words and this step is very useful because it removes a huge number of unused letters and it reduces the amount of text used by the classification algorithm. Then Arabic words will be normalized whereas some letters can be written in the other place. For example, a letter can" ا "be written instead of "ٳ" or "ٱ" so in this step the letters

[1] https://github.com/galaldev/ArabicNewsDatasets

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

that can be written instead of each other will be replaced. The Table 1 illustrates the letters that are replaced.

Table 1 Examples of Arabic letters normalization

Original Letter	Replacement Letter	Example of Original word	Example of Replacement word
ا, أ, إ, آ	ا	أحمد	احمد
ي, ى	ي	بنى	بني
ؤ, و	و	مؤمن	مومن
ة, ه	ة	الحديقه	الحديقة

We also remove Arabic diacritics such as (ً , ٍ , ٌ , ّ , ْ , ـَ,ـُـِ) because in the Arabic language the same word can be written with different diacritics according to its position on the sentence and diacritics are not used in extracting the Arabic word roots and useless in the classification task. After that, all words with length of less than three letters are considered trivial and doesn't affect the classification accuracy [10], so it will be removed.

3.2 Convert Text to Vectors

In this step Text inputs will be converted to vector and Perhaps the common way to do this is to use the BOW which done by representing the frequency of all the distinct words that are present in the document or bag-of-n-grams (BON) which is similar to BOW but uses n-gram which is a collection of n successive words [30], because it is very simple but it has many disadvantages such as different sentences may have the same representation, high dimensionality and ignore word semantics for example words like "Cairo," "Alexandria" and "Cooker" may be equally distant but semantically "Cairo" should be closer to "Alexandria" than "Cooker." , So we didn't use these techniques in this research [22].

distributed representations of words in a vector space can help algorithms to be more efficient by grouping the nearest words in the semantic which can be represented with near vectors. WE gives us the ability to generate close vectors for close words in the semantic and the very interesting thing this gives us the ability to make calculations on the words by their vectors for Example vec("Madrid") - vec("Spain") + vec("France") is closer to vec("Paris") than to any other word vector [28]. So WE can achieve better performance in ARTC tasks than the traditional BOW [20].

Word2Vec [18], [28] is one from popular word embedding algorithms recently which used to construct vector representations for words. To do convert words to vector space, it uses a shallow unsupervised ANN and takes each sentence and encodes the context of each of the words in the sentence. It learns document and let each word to learn other words into the ANN. Words of the associated meanings have a high likelihood to appear nearby on the document and two words are getting closer vector by repeatedly learning [31]. There are two types of architectures used by Word2Vec. The first called Continuous Bag-of-Words (CBOW) which trained to predict the current word by it's surrounding context. The second called Skip-Gram (SG) which trained to predict the Context by using a given word [26].

So, to convert text to vector space Word2Vec has been used to learn word vectors, after learning the word vectors we represent each document as a 2D matrix each row of this matrix is a single word vector as illustrated in Figure 2.

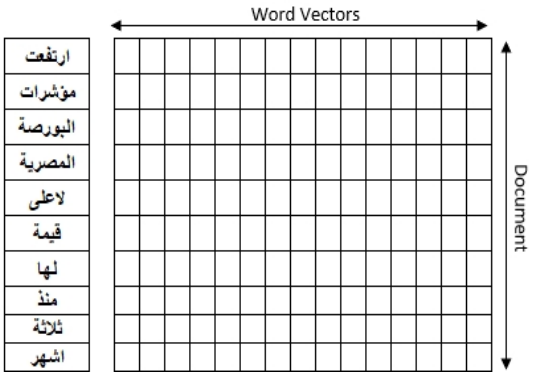

Figure 2 Example of Arabic 2D Document matrix

In the next step, we propose a new algorithm to statistically stem Arabic words by grouping similar words based on Arabic extra letters between words and the cosine similarity (CS) between words vectors then we named it with GStem.

3.3 GStem - Grouping similar words based on extra letters

In the Arabic language, there are 10 letters defined as Extra Letters which is extra to word root and these letters are (السين الهمزة اللام التاء الميم الواو النون الياء الهاء الالف) For example the root word "قال" which means "said" can be represented in future present by "سيقول" after adding Extra Letters "سيـ" at the beginning [32]. The problem with these letters it inserted to word root and give different words with the same meaning and the classification algorithm deals each word as a new distinct word. These Extra letters may be added

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

as prefix to the beginning of the word such as the previous example or added as suffix such as "قال" (Said) and "قالوا" (They said) or inserted inside the word stem such as "قال" (Said) and "قول" (Speech) and our GStem algorithm can deal with these all types of affixes.

This algorithm aims to group similar words sharing the same root (based on Arabic language extra letters) and semantically near to each other's then represent each group with only one word. Table 2 illustrates some examples.

Table 2 Examples of similarity groups based on Arabic Extra Letters

Words in the group	Group name
قلت,وقال,يقول,وقالت,سيقول	قال
عاما,عامان,اعوام,العام	عام
يحقق,حققت,تحقيق,تحقق	حقق

At first, we need to find a method to compute the string similarity distance between two given words based on insert or delete or substitute some of the Arabic extra letters and the best method for doing that is to compute the edit distance (ED) which is the minimum number of editing operations (Insertions, Deletion and Substitution) needed to transform one word into another. For example, to transform from "good" to "bold" there are 2 operations, substitute the g with a n b and substitute one o with al [33].

In this research, we use Alphabet-weight edit distance or we can call it the weighted edit distance (WED) by weighting the Arabic extra letters with weights less than the other letters [34]. For example if we compute the ED between two words "قال" (Said) and "سيقول" (Will Say) by the default ED algorithm it will result **2** because the word "قال" (Said) inserted by two letters "سي" to converted to the word "سيقول" (Will Say). However, if we compute the ED between two words "قال" (Said) and "قاد" (Lead) it will result in **1** because the word "قال" (Said) substituted on letter "ل" and replace it with "د" to converted to the word " قاد " (Lead). Then the word "قال" (Said) give a higher ED with "سيقول" (Will Say) which share the same root and a less ED with the word " قاد " (Lead) which has different meaning and root.

To avoid the previous problem, we use a WED algorithm which gives us the ability to give a weight for each insert, delete and substitute for each letter. Then we give a small distance for example **0.1** for all Arabic extra letters and **1** for other letters. If we recomputed the distance between two words "قال" (Said) and "سيقول" (Will Say) it will result in **0.1** which is less than the distance between two words "قال" (Said) and "قاد" (Lead) which gives **1.** Then the result of this string similarity function based on Arabic extra letters weights it will give low distance with words can extract from each other's by insert or delete or substitute some of the Arabic extra letters.

In Arabic language may one word give another meaning but give a lower WED because the word root itself contains a letter or more from extra letters Such as the word "قال" (Said) and "قتل" (Killed) it will give **0.1** WED because it will only substitute the letter "ا" with the letter "ت" and both letters are from Arabic extra letters. So, to solve this problem we computed the distance between the two words vectors which computed by the Word2Vec algorithm in the previous step of this model architecture. Whereas the Word2Vec let words of the associated meanings have a high likelihood to appear nearby on the document and two words are getting closer vector by repeatedly learning [31], then the two different meaning words "قال" (Said) and "قتل" (Killed) will have a high distance between their word vectors. So, we use the CS function to calculate word vectors similarity.

So, to check if two words sharing the same root, we need two parameters first: the top count of nearby words to the input word based on CS of the word vectors (TCCS), second: the accepted WED (AWED) between the words. For example, in our experiments, we find the similarity between the top 200 words nearest to the input word based on CS of word vectors and we group the words based on 0.3 WED which will give just three extra letters differences. Figure 3 illustrates how our GStem algorithm work to find words similarity groups based on its extra letters and word semantic.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

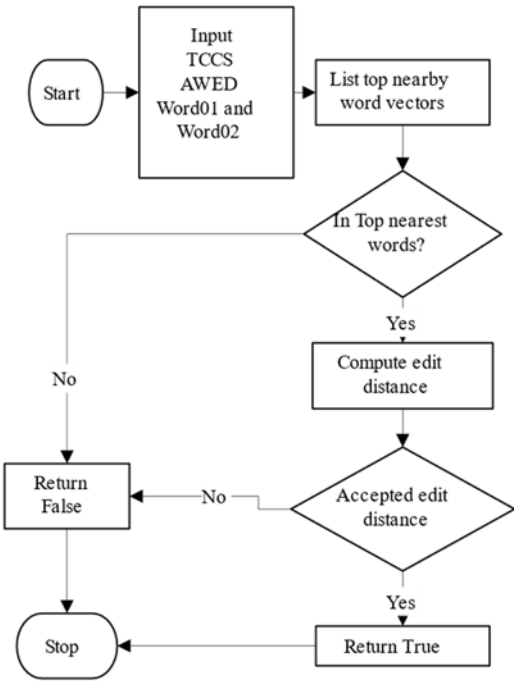

Figure 3 GStem function

We iterate each distinct word in the whole input text data then we compare each word with other words in the list and find if these words sharing the same root based on GStem function which will return True if the words related to each other otherwise will return False. The first step we pass **Word1** and **Word2** to find if these sharing the same root, **Top (count) nearest word vectors based on CS (TCCS)** and the **accepted WED (AWED)** Then list the top nearest words to **Word1** based on CS then check if the **Word2** exist in top list words nearest to **Word1** If the answer is yes then we compute the WED between the two words with respect to Arabic Extra Letters weights (0.1 for each letter in Arabic Extra Letter List) then check if the WED is less than or equals to the AWED parameter if the answer is yes then we consider the two words sharing the same root group. We repeated the previous steps on all given words recursively After that we can find groups of words sharing the same root then we replace each word in the input text with its root word (the smallest length word in the group) and produce a new stemmed input text.

3.4 Training

After representing documents words as vectors then we combine each word vector in each document and produce 2D matrix represent a single document. Then we pass the input documents matrixes to the CNN. The model architecture is a slight variant of the CNN architecture of Kim 2014 [1].

We build the model with a single channel input for WE. The convolution layer consists of several filters which are sliding window multiply its values element-wise with the input WE matrix, then sum them up. The primary purpose of Convolution is to extract features from the input matrix. After the convolution layer, a pooling layer applied to subsample their input by applying a max operation to the output of each filter. This gives a fixed size output matrix and reduces the output dimensionality of the output matrix. The outputs from previous layers are passed to a Fully Connected layer with SoftMax to use these features for classifying the input matrix into various classes based on the training dataset. We do regularization by using dropout on the SoftMax layer to prevents the co-adaptation of hidden units by randomly dropping out.

3.5 Model Evaluation

To evaluate the performance of the training models to check if they classify documents into the correct category, we use the Accuracy measure as the following.

$$\frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Where
TP: is the number of documents which assigned correctly to the current class
TN: is the number of documents which assigned correctly to the negative class
FP: is the number of documents which assigned incorrectly to the current class
FN: is the number of documents which belong to the current class but not assigned to it

4. RESULTS AND DISCUSSION

We make our experiments in two types of models. CNN-Norm which all word vectors learned from scratch using SG word2vec then learn document classes using the CNN without using GStem. CNN-GStem which all word vectors learned from scratch using SG word2vec then run GStem algorithm to group similar words based on their extra letters finally learn document classes using the CNN. We log the result of CNN-GStem model with some variations of GStem parameters as 0.3 and 0.4 for AWED and 200, 500 and 1000 for TCCS. We select model hyper-parameter by a grid search on the tested dataset. For word2vec we learn it using SG architecture, select 5 for window size,50 for vector

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

dimensionality and train it with 200 iterations. For CNN filter windows, we made it 2,3, 5 with 10 feature maps for each filter, 0.5 for dropout rate,50 for mini-batch and 20 learning epochs. We split the dataset as 80% for learning and 20% for testing and evaluation. The implementation of CNN is done by Keras [35], with TensorFlow backend [36].

Table 3 illustrates our model results based on the two model types with the variation of GStem parameters and the effect on distinct vocabulary count and the Accuracy measure.

Table 3 Model Results

Model Type	AWED	TCCS	V. Count	Acc
CNN-Norm	-	-	50,363	88.75 %
CNN-GStem	0.3	200	44,317	92.42 %
	0.3	500	40,946	89.92 %
	0.3	1000	37,562	88.75 %
	0.4	200	36,379	90.17 %
	0.4	500	30,578	88.83 %
	0.4	1000	26,517	87.50 %

Figure 4 focus on the change of accuracy per each model type.

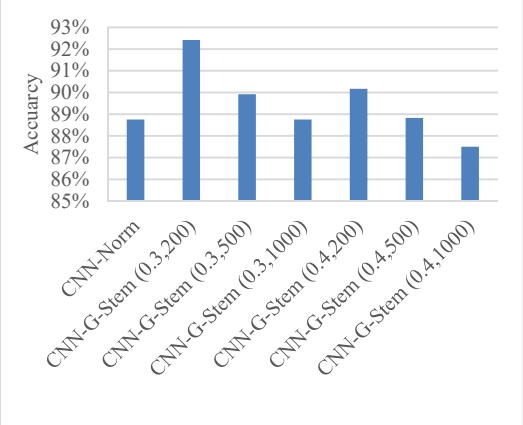

Figure 4 Accuracy per each model type

The result shows that the effect on Accuracy when using CNN-GStem outperform CNN-Norm. This improvement achieved because using the highest Accuracy GStem variation (0.3,200) gives the ability to group words share the same root and reduces the distinct vocabulary count to 44,317 while the original distinct vocabulary count is 50,363 which means it reduced by 12%. The result also shows that the most suitable AWED is 0.3 and this because in Arabic language root words with 3 extra letters or less is popular than with words with more than 3 extra letters. also, the most suitable TCCS is 200 but this parameter maybe varies according to the distinct vocabulary count of the training dataset.

We study the Accuracy measure change per epochs in Figure 5 which illustrates the change of Accuracy per epochs to reach the maximum value comparing CNN-Norm and CNN-GStem giving 0.3 for AWED and 1000 for TCCS parameters.

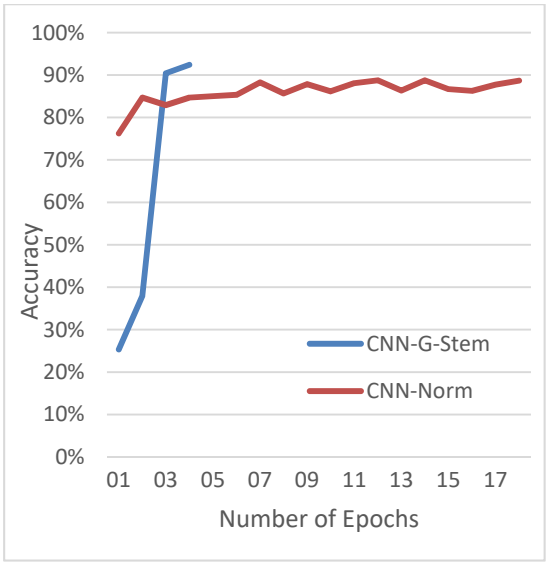

Figure 5 The increasing of Accuracy per Epochs

From Figure 5 we noticed that when using GStem it takes only 4 epochs to reach the maximum value of Accuracy measure and it takes 18 epochs to reach the maximum Accuracy value of CNN-Norm model which already less than the CNN-GStem Accuracy value and that's because GStem reduces the count of distinct words which helps the CNN to reach faster to its maximum Accuracy.

5. CONCLUSION

In this work, we introduced a new technique (GStem) to group similar words that share the same root based on the Arabic extra letters as a preprocessing layer for the CNN. In order to test our model, we collected an Arabic news dataset and used it in our experiments. We trained our WE from scratch depending on the collected dataset training samples and we didn't depend on any pre-trained data for word2vec or for GStem. Our experiments showed that when using GStem as a preprocessing step it increases the accuracy of the CNN model and this because the number of distinct words has been reduced. Our future plan is to adjust GStem to add special weights for most frequent letters and adjust it to deal with other extra letters such as prepositions attached to the word. We also planned to modify it to deal with another language.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6. REFERENCES

[1] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pp. 1746–1751, 2014.

[2] W. Yih, X. He, and C. Meek, "Semantic parsing for single-relation question answering," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2014, vol. 2, pp. 643–648.

[3] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[4] S. Alsaleem, "Automated Arabic Text Categorization Using SVM and NB," *International Arab Journal of e-Technology*, vol. 2, no. 2, pp. 124–128, 2011.

[5] S. Kaur and N. K. N. Khiva, "Online news classification using Deep Learning Technique," *International Research Journal of Engineering and Technology (IRJET)*, vol. 3, pp. 558–563, 2016.

[6] A. Noaman and S. Al-ghuribi, "A NEW APPROACH FOR ARABIC TEXT CLASSIFICATION USING LIGHT STEMMER AND PROBABILITIES," *INTERNATIONAL JOURNAL of ACADEMIC RESEARCH*, vol. 4, no. 3, pp. 114–122, 2012.

[7] M. El-Masri, N. Altrabsheh, and H. Mansour, "Successes and challenges of Arabic sentiment analysis research: a literature review," *Social Network Analysis and Mining*, vol. 7, no. 1, pp. 1–22, 2017.

[8] L. M. Al Qassem, D. Wang, Z. Al Mahmoud, H. Barada, A. Al-Rubaie, and N. I. Almoosa, "Automatic Arabic Summarization: A survey of methodologies and systems," *Procedia Computer Science*, vol. 117, pp. 10–18, 2017.

[9] M. Mustafa, A. S. Eldeen, S. Bani-Ahmad, and A. O. Elfaki, "A Comparative Survey on Arabic Stemming: Approaches and Challenges," *Intelligent Information Management*, vol. 09, no. 02, pp. 39–67, 2017.

[10] A. Ayedh, G. TAN, K. Alwesabi, and H. Rajeh, "The Effect of Preprocessing on Arabic Document Categorization," *Algorithms*, vol. 9, no. 2, 2016.

[11] S. Alowaidi, M. Saleh, and O. Abulnaja, "Semantic Sentiment Analysis of Arabic Texts," *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 8, no. 2, pp. 256–262, 2017.

[12] M. Al-Ayyoub, A. Nuseir, K. Alsmearat, Y. Jararweh, and B. Gupta, "Deep learning for Arabic NLP: A survey," *Journal of Computational Science*, 2017.

[13] R. Johnson and T. Zhang, "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks," *arXiv:1412.1058v2 [cs.CL]*, vol. 2, no. 2011, 2015.

[14] M. E. L. Kourdi, M. Elkourdi, A. Bensaid, and T. Rachidi, "Automatic Arabic Document Categorization Based on the Naïve Bayes Algorithm," *Proceedings of COLING 20th Workshop on Computational Approaches to Arabic Script-based Language*, pp. 51–58, 2004.

[15] H. M. Alserhan and A. S. Ayesh, "An application of neural network for extracting Arabic word roots," *WSEAS Transactions on Computers*, vol. 5, no. 11, pp. 2623–2627, 2006.

[16] R. Al-Shalabi and R. Obeidat, "Improving KNN Arabic Text Classification with N-Grams Based Document Indexing," *INFOS2008, March*, pp. 108–112, 2008.

[17] L. S. Larkey, L. Ballesteros, and M. E. Connell, "Light Stemming for Arabic Information Retrieval," in *Arabic Computational Morphology*, no. Ldc, Dordrecht: Springer Netherlands, 2007, pp. 221–243.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, vol. 1, pp. 1–12, 2013.

[19] A. Dahou, S. Xiong, J. Zhou, M. H. Haddoud, and P. Duan, "Word Embeddings and Convolutional Neural Network for Arabic Sentiment Classification," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp.

2418–2427.

[20] A. El Mahdaouy, E. Gaussier, and S. O. El Alaoui, "Arabic Text Classification Based on Word and Document Embeddings," in *International Conference on Advanced Intelligent Systems and Informatics. Springer, Cham*, vol. 533, 2017, pp. 32–41.

[21] J. P. and R. S. and C. D. Manning, "GloVe: Global Vectors for Word Representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, vol. 31, no. 6, pp. 1532–1543.

[22] Q. V. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," in *International Conference on Machine Learning*, 2014, vol. 32.

[23] R. Khoja, S., Garside, "Stemming arabic text," *Computer Science Department, Lancaster University, Lancaster, UK*, 1999.

[24] M. Saad, "The Impact of Text Preprocessing and Term Weighting on Arabic Text Classification," 2010.

[25] M. Saad and W. Ashour, *OSAC: Open Source Arabic Corpora*. 2010.

[26] S. Al-Azani and E.-S. M. El-Alfy, "Hybrid Deep Learning for Sentiment Polarity Determination of Arabic Microblogs," vol. 10638, D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, Eds. Cham: Springer International Publishing, 2017, pp. 491–500.

[27] S. Boukil, F. El Adnani, A. E. El Moutaouakkil, L. Cherrat, and M. Ezziyyani, "Arabic Stemming Techniques as Feature Extraction Applied in Arabic Text Classification," in *International Conference on Advanced Information Technology, Services and Systems. Springer, Cham*, vol. 2, 2018, pp. 349–361.

[28] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," pp. 1–9, 2013.

[29] W. Zaghouani, "Critical survey of the freely available Arabic corpora current situation of the freely available," *Proceedings of the Workshop on Free/Open-Source Arabic Corpora and Corpora Processing Tools Workshop Programme,LREC*, pp. 1–8, 2014.

[30] Z. S. Harris, "Distributional Structure," *WORD*, vol. 10, no. 2–3, pp. 146–162, Aug. 1954.

[31] C. A. Turner *et al.*, "Word2Vec inversion and traditional text classifiers for phenotyping lupus," *BMC Medical Informatics and Decision Making*, vol. 17, no. 1, p. 126, Dec. 2017.

[32] م. عبدالمعطي, *شذا العرف في* and ا. الحملاوي *فن الصرف*. 1957

[33] D. Jurafsky, "Minimum Edit Distance," 2012. [Online]. Available: https://web.stanford.edu/class/cs124/lec/med.pdf.

[34] D. Gusfield, "Algorithms on Strings, Trees, and Sequences," *Computer Science and Computational Biology. Cambridge University Press, Cambridge, UK*, p. 172, 1997.

[35] François Chollet, "Keras," *GitHub repository*, 2015.

[36] A. Agarwal *et al.*, "TensorFlow : Large-Scale Machine Learning on Heterogeneous Distributed Systems," 2015.