Egyptian Informatics Journal 22 (2021) 35–43

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Full length article

A simple Galois Power-of-Two real time embedding scheme for performing Arabic morphology deep learning tasks

Mohammed A. ELAffendi [a,*], Ibrahim Abuhaimed [b], Khawla AlRajhi [b]

[a] Department of Computer Science, CCIS, Prince Sultan University, Saudi Arabia
[b] EIAS Data Science & Blockchain Lab, CCIS, Prince Sultan University, Saudi Arabia

ARTICLE INFO

Article history:
Received 22 December 2019
Revised 8 March 2020
Accepted 29 March 2020
Available online 21 April 2020

Keywords:
Word embeddings
Deep learning
Arabic morphology
Galois power of two
Real-time embeddings
Parallel neural model graph

ABSTRACT

This paper describes how a simple novel Galois Power-of-Two (GPOW2) real-time embedding scheme is used to improve the performance and accuracy of downstream NLP tasks. GPOW2 computes embeddings live on the fly (real time) in the context of target NLP tasks without the need for tabulated pre-embeddings. One excellent feature of the method is the ability to capture multilevel embeddings in the same pass. It simultaneously computes character, word and sentence embeddings on the fly. GPOW2 has been derived in the context of attempts to improve the performance of the SWAM Arabic morphological engine, which is a multipurpose tool that supports segmentation, classification, POS tagging, spell checking, word embeddings, sematic search, among other tasks. SWAM is a pattern-oriented algorithm that relies on morphological patterns and POS tagging to perform NLP tasks. The paper demonstrates how GPOW2 led to improvements in the accuracy of POS tagging and pattern matching, and accordingly the performance of the whole engine. The accuracy for pattern prediction is 99.47% and is 98.80% for POS tagging.

1. Introduction

Distributional semantics and word embedding techniques [27,25,26] are currently the norm in performing deep learning NLP tasks of any kind. Although these techniques may be traced back to the work of Bengio et al. [3], they only became popular and attracted attention after the surprising success of the Word to Vector (Word2Vec) approach proposed by Mikolov [27,25,26]. Following the success of the Word2vec approach, many other superior approaches have been proposed, including Elmo [32], UMLfit [19] and BERT [8], which capitalized on the power of Bidirectional LSTM models, Transfer Learning, transformer and attention techniques.

Despite the success of word embedding approaches and the breakthroughs they facilitated, there are still some areas of improvement, which include:

1. The required distributional semantic vectors are provided as universal pre-computed tables generated from large corpora. Due to the heavy computational load, the process cannot be integrated real-time in NLP models
2. As may be expected, embedding vectors are only computed for words that appear in the corpus. Out of vocabulary words are not represented in the computed tables
3. Computing the pre-embeddings is a resource intensive process in terms of compute time and memory
4. Since the sematic vectors are computed over the whole corpus (in most approaches), local contexts of words may be negatively overshadowed by the corpus global contexts. Although this problem has been addressed by new approaches such as Elmo [32], UMLfit [19] and BERT [8], the resulting models are still complex and difficult to integrate.
5. Most of the current approaches do not easily scale up or down to compute sentence or character embeddings

This paper describes a simple real-time neuro embedding scheme that computes embeddings live on the fly in the context of the target NLP task, without the need for tabulated pre-embeddings or pre-trained transfer learning models.

* Corresponding author.
E-mail addresses: affendi@psu.edu.sa (M.A. ELAffendi), iabuhoimed@psu.edu.sa (I. Abuhaimed), khrajahi@psu.edu.sa (K. AlRajhi).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

https://doi.org/10.1016/j.eij.2020.03.002
1110-8665/

The main features and advantages of the new scheme include the following:

1. The scheme is based on a simple Galois Power-of-Two (GPOW2) integer transformation that preserves the positions of characters and words in the text embedding vectors. For example, in the case of base-64, each character is represented by six bits in the resulting embedding vector.
2. Being a Galois transformation, modular arithmetic can be used to simplify operations in case of large vectors
3. The scheme easily scales up and down to generate sentence and character embeddings. Character-word-sentence embeddings (Triple-Embed) may be applied in parallel to perform NLP tasks
4. The embeddings are generated real-time as part of the training process, without the need for tabulated embeddings
5. As will be demonstrated below, GPOW2 leads to high accuracies that compare favorably with some of the established baselines

The power of GPOW2 is illustrated by applying the method to a real-life problem in Arabic morphology. The SWAM morphological engine [17] grew over the years from a simple morphological analyzer to a multipurpose linguistic tool that addresses spelling correction, sematic search, sentence embedding, plus many other tasks. The algorithm is pattern-oriented and applies a traditional probabilistic language model to capture the contexts of words and morphemes. The main problem of the algorithm is that it is relatively slow due to the exhaustive search process for the appropriate pattern. Replacing traditional probabilistic language models with GPOW2 to predict word patterns and pos tags substantially improved the performance and accuracy of SWAM.

The paper is organized as follows: Section 2 below states the main problem. The literature review is given in Section 3. Section 4 introduces the novel Galois Power of Two (GPOW2) reversible text transformation and demonstrates how it works. Section 5 describes how GPOW2 embeddings are directly integrated in neural models that predict NLP targets. The neural model is described in detail in this section. Results of experimentation and findings are given in section 6. Section 7 provides a detailed example illustrating how the model works. Section 8 shows how the proposed neural model substantially improves the performance of traditional SWAM. Conclusions and recommendations are presented in the final section.

2. The problem

As mentioned in the introduction, most embedding schemes provide their services through large lookup tables wherein each word is represented by only one embedding vector computed over a large corpus. The problems with these approaches have been outlined in the introduction.

2.1. Research objectives

The main objectives of this study include

- Develop a simple universal embedding scheme that computes real-time contextual embeddings on the fly in the context of the target NLP task.
- Solve the scalability problem of current embedding approaches and provide guidance as how to compute embeddings at character, word, and sentence levels.
- Replace the SWAM exhaustive search processes by a one-step neural prediction model to improve performance

3. Literature review and related work

Distributional semantics and vector space representations can be traced back to the 1990's, where Latent Semantics Analysis (LSA) [7,34] and Latent Dirichlet Analysis (LDA) [4] have been used in information retrieval and topic modeling. However, Word embeddings as we know them today originated in the work of Bengio et al. [3] who are the first to use a neural model to learn distributional representations of vocabulary words based on their co-occurrences in a corpus. Their model performs the following:

1. Each word in the vocabulary is associated with a continuous dense feature vector
2. The continuous feature vectors are generated using a neural network model, taking into consideration the sequence of occurrences in the corpus (context of the word)
3. The joint conditional probability for the predicted word is computed using a Softmax activation function (as is the common practice in current word embedding models)
4. The model includes at least one hidden layer, representing the feature vectors

One of the interesting observations of Bengio et al is that the distributional representations exhibit a notable generalization capability and are better in handling out-of-vocabulary words.

Although the results and findings of Bengio et al. were very interesting, research in the area stagnated for some time, presumably due to the lack of appropriate computational power at the time. The next milestone in the evolution of word embeddings is the work of Collobert and Weston [6] who developed a unified multitask neural model, which simultaneously performs multiple NLP tasks, based on the distributional feature vectors computed using a deep learning model, similar to the one developed by Bengio et al.

The main difference between the work of Bengio and Collobert (apart from the ability to perform multiple NLP tasks) is the use of convolutional neural networks in Collobert's study, instead of the common MLP's used in the work of Bengio.

The third and, by far, the most important milestone in the evolution of word embeddings is the work of Mikolov et al. (Mikolov et al., 2014; Mikolov et al., 2016) who developed the Word to Vector (Word2Vec) approach to compute distributional word representations. The main difference between the Word2Vec and the previous approaches is the use of shallow neural models to learn the vectors, which made it feasible to train the model on large corpora in a short time. Another difference is the use of a symmetric context window around the focus word.

The Word2Vec approach comes in two flavors: Continuous Bag of Words (CBOW) and Skip-gram. In the CBOW version, a semi-supervised neural network is used to compute the embedding vector for a given word. In the Skip-gram version, the Skip-gram model is used to predict the neighbors of a given word.

The fourth wave in the evolution of word embeddings is "transfer learning" [19,32], where pretrained deep learning word embedding models are used to boost the performance and accuracy of prediction and classification models. Instead of starting from scratch, classification models can reuse deeply trained models on similar tasks to generate more accurate results in less time. The practice has been borrowed from the image recognition field where pretrained models are used to train new models producing better accuracy and performance. Good examples of transfer learning include ELMo [32] and UMLfit [19].

Recently, some efforts have been made to improve the Word2Vec and extend it to cover sentence embeddings. These efforts include the works of Ksuner et al. [22] and Mijangos, Sierra, and

Herrera [24] to compute sentence embeddings and the work of Joulin et al. [20] to include character embeddings as well as word embeddings.

The work of Mikolov has also been extended to cover sentence embedding [26,27], document embedding [28,30,22], and character embedding [20]. This made it possible to incorporate the technology into many applications and production settings. A good example is the Fast-text system that integrates character embedding with word embeddings [20].

Many efforts have been reported regarding the application of word embeddings to perform core NLP tasks. The most relevant tasks in our case include the work of Zhang, Yu, and Fu in Chinese sentence segmentation [36].

3.1. Arabic word embeddings and applications

Altowayan and Tao [2] used word embeddings for Arabic sentiment analysis. They started by compiling a large Arabic corpus from various sources, based on which they computed word embedding vectors for both standard and dialectic Arabic. The resulting distributional word representations have then been used to train binary sentiment classifiers to determine subjectivity and sentiment. The main motivation of the authors was to avoid manual feature engineering, which is a difficult time-consuming task. They reported that their results are slightly better than the traditional approaches based on handcrafted features.

Dahou et al. [9] used a relatively big corpus to generate Arabic word embeddings using both CBOW and Skipgram models. The authors tested and compared the quality of the generated word embeddings with results generated by similar and traditional approaches.

The SWAM Arabic morphological analysis algorithm [15,17] is a member of a family of emerging Arabic morphological tools that include BAMA [5], MADA + TOKEN [18]and MADAMIRA [31].

The application of deep learning and neural network techniques to Arabic morphology are well covered in ELAffendi and Alrajhi [14,16], who also introduced an interesting RIT64 text encoding alternative to one-hot encoding that turned out to be faster by many orders of magnitude and lead to more accurate results.

4. A novel reversible Galois Power-of-Two (GPOW2) text embedding scheme: a simple tweak that works

The models used in this paper and the good results obtained are based on a simple reversible Galois Power-of-Two (GPOW2) integer transformation through which all input words are encoded as polynomials (summations) of power of two integers. GPOW2 is a generalization of the RIT64 transformation [14] and, as implied by the name, capitalizes on the properties of Galois finite fields [23,33] to perform a variety of algebraic operations on the integer representations of words.

The transformation is based on the simple idea that words and strings may be viewed as integers in a number system whose radix r is a power of two. The characters used to form words and strings are the digits of this number system. The radix r should be large enough to accommodate all characters used to form words and strings. In the current version of GPOW2, the value of r is 64 (2^6), which accommodates up to 64 characters. Based on this assumption, each word or string of length m is represented as a polynomial:

$$W = a_m r^m + a_{m-1} r^{m-1} + \ldots + a_1 r^1 + a_0 r^0$$

where the a_i's here represent the digits, and r is the radix

4.1. What is a Galois Field

A Galois or Finite Field F is defined as follows [23,33]:
A Galois or Finite Field F is a finite set of elements with the following properties:

- All elements of F form an additive group with the group operation "+" and the neutral element 0.
- All elements of F except 0 form a multiplicative group with the group operation " x" and the neutral element 1.
- When the two group operations are mixed, the distributivity law holds:
  for all a, b, c ∈ F: a(b + c)= (ab)+ (ac).

The number of elements in the field is called the order and cardinality of the field. A Galois field of order m exists if m is a prime power: $m = p^n$, for some integer n and a prime p. In this case p is the characteristic of the field

4.2. Galois prime fields GF(p)

A Special case of Galois Finite Fields are Galois Prime Fields referred to as GF(p). A Galois Prime Field is a finite field with a prime number of elements, i.e. m = p, for some prime p (implying n = 1 in the above relation). In GF(p) every nonzero element has an inverse and arithmetic is done modulo p.

4.3. Why do we care?

Galois fields have been successfully used in cryptography [23] and communication channel coding [33]. It appears from our initial experiments that Galois finite fields will have some impact in the areas of text encoding and text processing. The main attraction is that we can define two operations + and * corresponding to two abelian groups. One of the implications of Galois finite fields is the possibility of using modular arithmetic to perform embedding operations for sentences and large documents.

4.4. Galois prime fields for Arabic words and strings

Because Arabic words can be represented as integers base power-of-2 (the radix of the resulting integer is a power of 2), we can easily identify a Galois prime field for all words in the language, or multiple fields based on the length of the words using the Mersenne prime property associated with powers of two. A Mersenne prime is a prime number that is one less than a power of two, usually expressed as a number of the form $M_n = 2^n - 1$ for some integer n.

4.5. The power-of-two transformation

1. Every word is represented by a unique number whose radix r is a power of 2 (2, 4, 16, 32, ...).
2. The number of binary digits needed to represent each character is equal to the base-2 exponent n of the radix (in case of r = 64, the number of binary digits needed to represent a character is 6 (sine $64 = 2^6$)).
3. The resulting integer transformations are members of the Galois prime field [23] whose order is the upper closest Mersenne prime [33] to the sequence.
4. The above point implies that many string operations are modeled and executed within a Galois prime field. A Galois prime field is a field whose order is a prime p. Arithmetic is done modulo p.
5. The transformation is reversible since the radix of the resulting integers is a power of 2.

6. Because the radix is a power of 2, the transformation preserves the positions and order of characters in the binary representation of the word. In case of r = 64, each character is represented by six bits. This feature is the main enabler for superimposed character-word embedding.
7. The transformation is adaptive. The maximum number of language characters accommodated by this transformation is equal to the value of the radix. If more characters are needed, then one can easily switch to a higher power of 2 radix. For example, in the case of Arabic, it was realized that a maximum of 64 characters are needed to represent most strings in the language. This includes the 28-letter characters, some punctuation marks, and special symbols.

Detailed explanations and justifications are provided in [14,13].

5. A universal GPOW2 embedding scheme: a real-time simultaneous embedding model

To illustrate the power of GPOW2, a multichannel neural network model is used to predict pos tags and patterns of Arabic words. The model comprises three channels: char channel, word channel and sentence channel (Fig. 1 below). GPOW2 is used to compute the embeddings for the three channels (character embeddings, word embeddings and sentence embeddings). A convolutional neural model has been used in each of the word and character channels. For the sentence channel an LSTM has been used. The parameters of the models are described in Table 2 below.

6. Model, experimentation and results

The multichannel neural GPOW2 model described in section 5 above is applied in this section to a gold standard public Arabic morphological dataset to demonstrate its accuracy. The main purpose is to compute predictions for Arabic word patterns and POS tags. The dataset is described in section 6.1 below, while the results are given in section 6.2 Fig. 3.

6.1. The data set

Over the past few years, some efforts have been made to prepare fully annotated Arabic data sets and treebanks. These sets differ in their size, depth, and, more seriously, in the tag sets used. It is unfortunate that there is no agreement on a standard tag set for Arabic. A wide range of tag sets have been proposed [35]. The num-

Table 2
The Multichannel GPOW2 Model Details.

channel	Type of model	embedding	Comments
1	Bidirectional LSTM with 4 layers (backbone of the multichannel model (5 layers, including the concatenation layer)	GPOW2 sentence embedding	Output merged with the concatenation layer
2	Convolutional Neural Network (6 layers, two outputs)	GPOW2 word embeddings	Second output merged with the concatenation layer
3	Convolutional Neural network (6 layers, two outputs)	GPOW2 character embedding	Second output merged with the concatenation layer

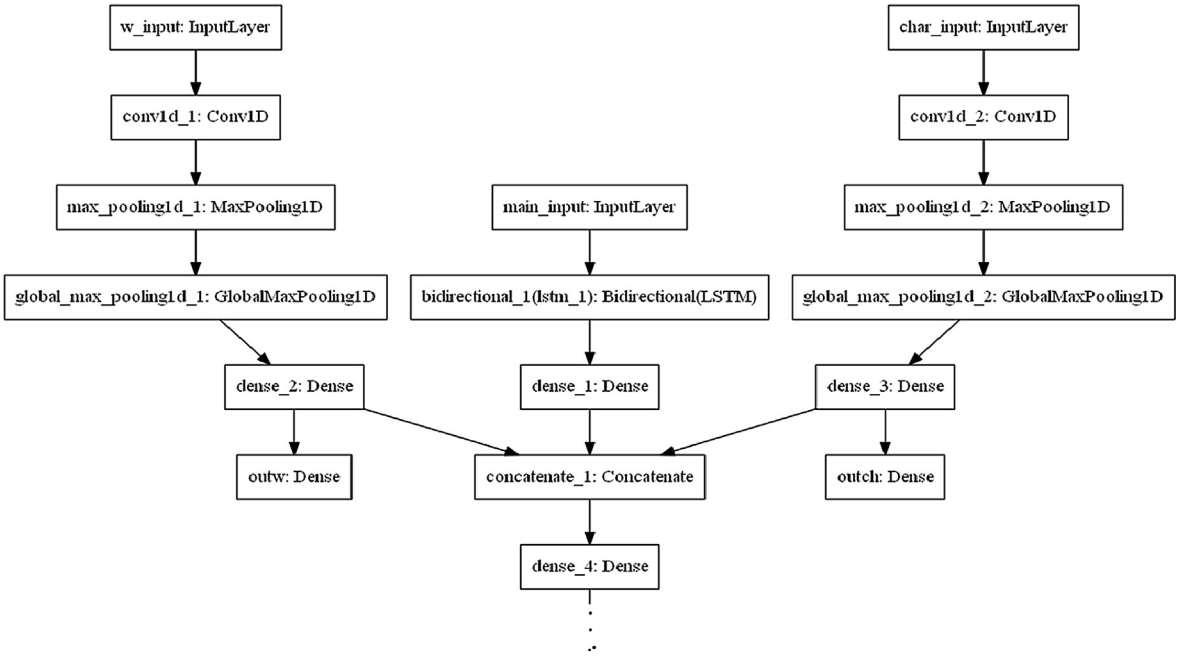

Fig. 1. The Multichannel GPOW2 Model with Three Inputs: Char, Word, Sentence.

Table 1
Binary Representation Resulting from RIT64 Forward Transformation for the word (المستقيم).

م	ي	ق	ت	س	م	ل	ا
100,100	101,001	100,001	001,001	010,010	100,100	100,011	000,110

ber of tags in each set varies from about 20 tags to more than 150. For example, the Reduced Tag Set (RTS) adopted by the LDA at Penn State University consists only of 25 tags, which, according to some authors [10] are not enough to cover all features of Arabic, while the tag set proposed by Khoja [21] contains about 171 tags.

The major determining criteria in choosing the data set for this study include:

1. The set should be fully annotated
2. Publicly available for researchers
3. The tag set used should be broad enough to cover standard Arabic features (not too small, not too detailed)

Based on these criteria and the comparisons made, the clear choice was the Quranic Arabic Corpus (QAC) prepared by Dukes [11,12] in the context of his work on Statistical Parsing at Leeds University. The main reasons for choosing QAC are:

1. The tag set used is a medium sized representative tag set that covers all standard Arabic features.
2. A transparent crowd sourcing process was used to annotate the corpus with many revision cycles.
3. The set is well-served, publicly available, and the quality of annotation is very high.
4. Detailed morphological-syntactic analysis has been provided for every word in the treebank.
5. The set is publicly available free to researchers on the project website http://corpus.quran.com/download/.

6.1.1. Our work on the dataset

As mentioned above, QAC provided very detailed segmentation and valuable syntactic and morphological information for every word in the corpus, including affixes, clitics, and all types of morphemes. However, some preprocessing was necessary to render the corpus usable in the context of the planned analysis. Accordingly, the following major preprocessing tasks have been performed:

1. Derive the morphological patterns (morphological templates) for all words in QAC since these were not provided in the original annotated set. A small tool was developed to derive the patterns. This is important, since SWAM is pattern oriented.
2. Extracting appropriate training and test tables for the respective tasks and models and decoding the compact format in which the corpus annotation is presented.
3. Computing the frequency and bigrams for all words and morphemes in the set. As noted above, the original SWAM uses statistical language models at the morpheme and word levels.

6.2. Experimentation and results

Two main experiments were performed to evaluate the accuracy and generalization power of GPOW2 embeddings. In both experiments, the multichannel GPOW2 model has been used to predict the respective word classes (POS or Pattern). Three parallel embedding models (three channels) have been concatenated into a unified embedding layer to produce the required predictions.

The three embedding channels are:

1. The char embedding channel which computes character embeddings using a convolutional model and the GPOW2 representation for each character (6 bits)
2. The word embedding model which computes word embeddings using a convolutional network and GPOW2 vector representation for each isolated word in the set (84 bits)
3. The sentence embedding model computes sentence embeddings (for the sentence in which the word occurs) using the word context vector representation for the sentence using a window of size 5 (each word with a window of size 5 (2 words before and 2 words after) (total 420 bits)

The set has been divided into an 80% training set and a 20% testing set. The tables below provide the details of the model parameters and accuracy for both experiments.

6.2.1. Experiment 1: Predicting word patterns using the GPOW2

The first experiment focused on predicting word patterns, which is the main performance problem in SWAM. The total size of the dataset is 77915, 80% of which has been used for training and 20% for testing. The other network parameters are given in Table 3 below. As apparent from the table, the accuracy is 99.24%. Fig. 2 provides a plot of the accuracy as a fnction of the number of epochs.

6.2.2. Experiment 2: Predicting POS tags using GPOW2

A similar setup was used to predict the part of speech tags for the same set. Table 4 below shows the parameters and accuracy of the POS tag experiment. Fig. 3 provides a plot of the accuracy as a function of the number of trainings epochs.

7. How it works: a detailed example

This section provides a detailed example to explain how the GPOW2 deep learning model works. The model uses a backpropagation algorithm to learn during the multi-epoch training process. A typical epoch (round) in the training process is shown in Fig. 4 below. The diagram represents one pass in the training process of a POS tagging model. The process may be summarized as follows:

1. To train the model to predict pos tags for a given input word, we need to compute the GPOW2 embedding for that word at the input level of the neural network (the first two bottom layers in Fig. 4).
2. For this purpose, we need to specify the size of the "context window" for the input word (a section of the input data in which the word occurs in the middle surrounded by n words on both sides).
3. In Fig. 4 below, the target word is "الصراط", and the size of the window is 3.
4. Accordingly, the GPOW2 representations of the three words in the window are passed to the neural network in the bottom layer as shown in Fig. 4 below.

Table 3
Parameters for the Pattern Prediction Model.

Parameter	Value
Total Set Size	77,915 items
Vocabulary size	7508
Text Encoding	GPOW2 encoding for input and output
Initial Training Set Size	80%
Initial Testing Set Size	20%
Target vector size	84 neurons (GPOW2 representation of patterns)
No of parallel input layers	Three (char, word, context of word using GPOW2)
No of output neurons	84 (using GPOW2)
Optimization	ADAMS
Activation Function	Sigmoid
Loss Function	Contrastive loss
No of Epochs	30
Train accuracy	99.47%
Validation acc	99.25%
Test Accuracy	**99.24%**

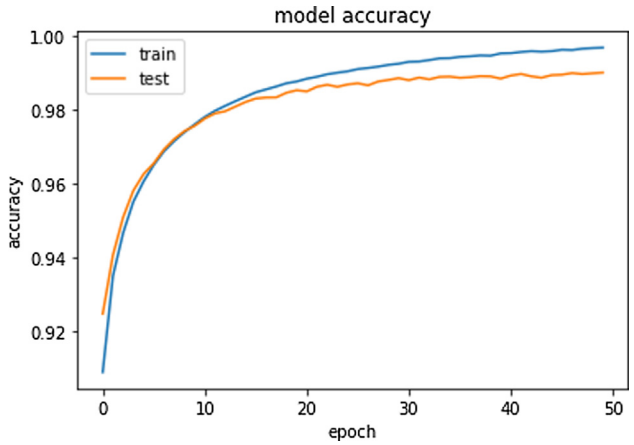

Fig. 2. The accuracy of pattern prediction.

Table 4
Parameters for the POS Prediction Model.

Parameter	Value
Total Set Size	77,915
Vocabulary size	7509
Initial Training Set Size	80%
Initial Testing Set Size	20%
Target layer size	84 neurons (GPOW2 representation of POS tags)
No of parallel input layers	Three (char, word, context of word using GPOF2)
No of output neurons	84 (using GPOW2)
Optimization	ADAMS
Activation Function	Sigmoid
Loss Function	Contrastive loss
No of Epochs	30
Train accuracy	99.48%
Validation acc	98.82%
Test Accuracy	**98.80%**

5. Based on these vectors, the real time embedding for the word is computed in the second layer from the bottom (Hidden Layer 1 in Fig. 4)
6. The embedding vector is then processed in the subsequent layers until it produces the "prediction" for the pos tag in the output layer (second from top if Fig. 4)
7. The resulting "prediction" is then compared to the GPOW2 representation of the actual tag (top box in Fig. 4) using the contrastive loss function.
8. The contrastive loss function computes the difference between the "prediction" (coming from the output layer) and the actual pos tag GPOW2 representation. The difference (loss) is then fed back to the neural network to compute another prediction in the next epoch
9. This goes on until an acceptable accuracy is reached through the minimization of the loss.

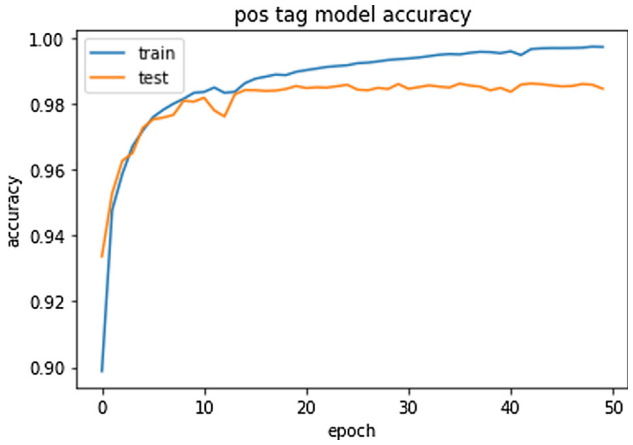

Fig. 3. The accuracy of POS tags.

Note that the GPOW2 representations for words and tags are computed using the following method:

```
def word2Bin64(word, binList):
    n64 = 0;
    # print("length =",length(word))
    for ch in word:
        #print(ord(ch)-1569)
        k=ord(ch)-1560
        if (k>=0):
            n64=n64*64+k
        wbin =bin(n64)
        #convert RIT64 encoding into a list of binary digits for neurons
        wbinList = [int(x) for x in wbin[2:]]  # create binary vector for word
        Lead = [0]*(84 -len(wbinList)) # add leading zero elements
        wbList =Lead +wbinList # add leading zeros to input layer vector to make it 60
        for x in wbList:
            binList.append(x)
      return(binList)
```

A simple example for how the GPOW2 representations of words, tags and patterns has already been given in Table 1 above.

7.1. Evaluations and comparisons with gold standard baselines and reported results

In this section our results are compared with established Gold Standard QAC baselines (Conventional SWAM), Buckwalter [13], Alashqar [1], and established baselines using LDC Arabic Tree Bank [29]. Table 5 below summarizes these baselines while the comparsins are shown in Table 6:

7.2. Analysis

1- Tables above shows compares pos tagging accuracies obtained using established baselines with our GPOW2 model results. The GPOW2 accuracies are given at the bottom of the tables in **bold**.
2- The POS tagging accuracy obtained using the word embedding neuro models (98.8%) clearly beats all baselines, including conventional SWAM gold standard.
3- The comparisons confirm that conventional SWAM is highly accurate and its only problem was the computational cost

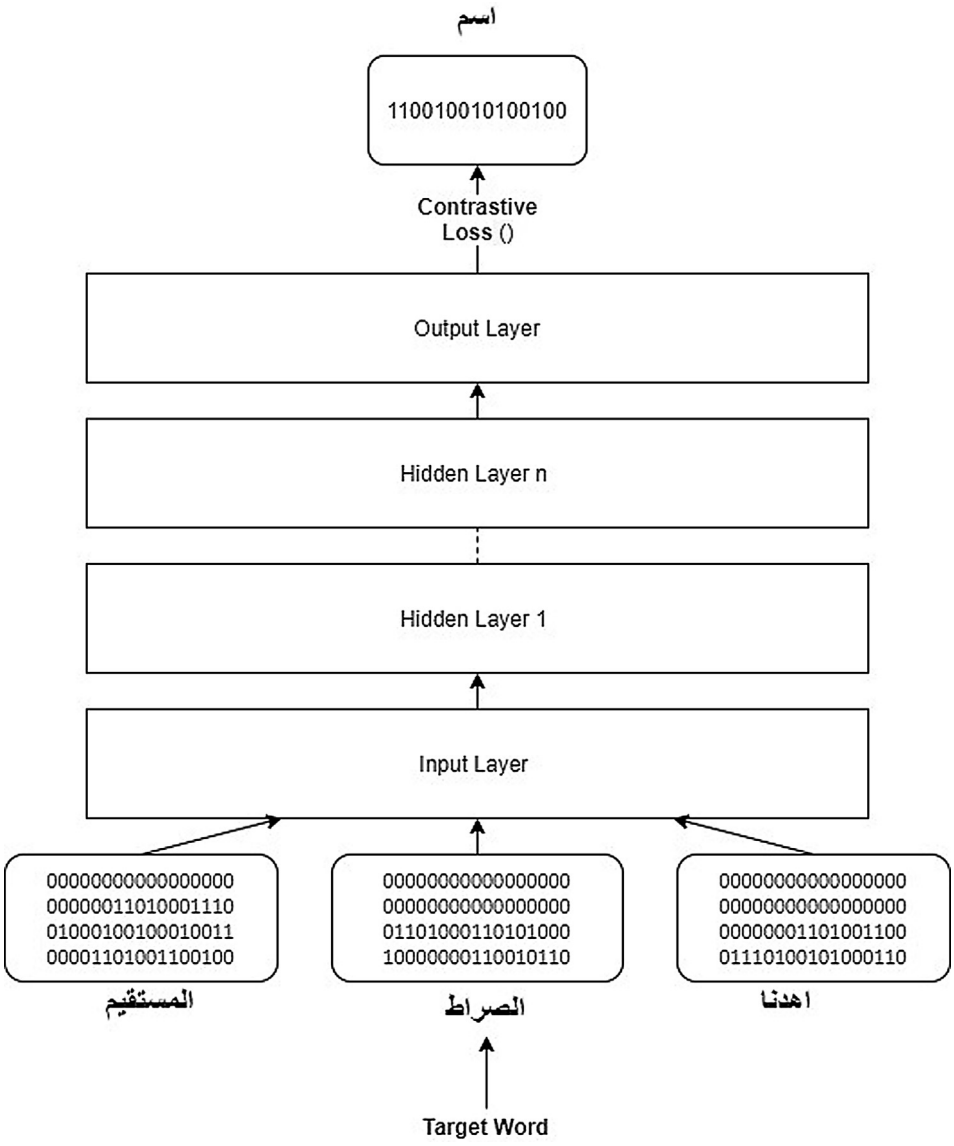

Fig. 4. A Simple Example of a Typical Channel in the GPOW2 Model.

Table 5
The baselines against which GPOW2 has been benchmarked.

Baseline	Description
Alashqar [1] 33 QAC tags	Alashqar [1] used 33 out of the QAC to compute the accuracy of six traditional tagging algorithms
Raw Original Swam- full tag set (Gold Standard)	[13] Applied a traditional language model approach to compute the accuracy of POS tagging and morphological pattern predictions
Buckwalter – full QAC tag set	Buckwalter [5] algorithm has been adapted to compute (predict) POS QAC POS tagsing using a subset of the LDC Arabic Treebank consisting of 500,000 dataset to compute
LDC pos tagging (LDC tag set)	[29] POS tagging using a subset of the LDC Arabic Treebank consisting of 500,000 dataset to compute
LDC segmentation accuracy	[29] POS tagging using a subset of the LDC Arabic Treebank consisting of 500,000 dataset to compute

4- Although segmentation was not directly addressed in the embedding models, pattern predictions accuracy (99.94%) is an indirect indication for segmentation accuracy. In SWAM, the pattern encodes the segmentation. Accordingly, the segmentation accuracy in Neuro SWAM also beats all baselines
5- The experiments performed by Alshqar (2012) shows that the size of the tag set has little effect on the accuracy. He used two sizes 33 and 9. In all our experiments we used the full tag set, around 42.

8. Neuro SWAM improvements over conventional SWAM

As expected, replacing the exhaustive search component in the original SWAM with a neuro predictor reduced the execution time by 93.11% while maintaining the same level of accuracy. Both versions of SWAM were applied to analyze the full QAC set. It took original iterative SWAM 1018.8 s to analyze the full set, while Neuro SWAM finished the full task in 70.2 s. The test was performed on an iCore 7 Laptop with 16 Gig RAM.

The accuracy remained almost the same. Original SWAM achieved an overall accuracy of 97.57% while the Neuro version achieved 96.8% in the tests performed. Table 7 below summarizes the results.

Table 6
Comparing GPOW2 against the baselines summarized in Table 5.

Experiment type	Accuracy of Taggers						Comparison Column
	Unigram	Bigram	Trigram	Brill	HMM	TnT	
Alashqar [1] 33 QAC tags	80.4%	80.5%	80.3%	80.9%	75.2%	69.2%	80.90%
Raw Original Swam- full tag set (Gold Standard)							98.43%
Buckwalter – full QAC tag set							92.68%
LDC pos tagging (LDC tag set)							94.60%
LDC segmentation accuracy							94.91%
GPOW2 neural model pos tagging							**98.80%**
GPOW2 neural model pattern classification acc							**99.24%**

Table 7
Performance of Original SWAM versus Hybrid Neuro SWAM using the QAC Data Set.

Parameter	Original SWAM Toolkit	Hybrid Neuro SWAM	Comments
Performance (using the full data set)	1018.8 sec	70.2 sec	93.11% performance improvement
Over all accuracy	97.57%	96.8%	Accuracy slightly less than original SWAM, expected to improve when more training is performed using larger data sets

9. Conclusion

A simple novel Galois Power of Two (GPOW2) scheme for simultaneously computing real time character, word and sentence embeddings has been introduced. The GPOW2 embedding approach is based on a simple power of two transformation that preserves the positions of characters and words in the resulting transformation of a sentence or document. This interesting feature made it possible to design parallel embedding graphs of varying complexity and depth.

To demonstrate the power of GPOW2, the method has been applied to solve a real life Arabic morphological problem related to the performance of the SWAM linguistic tool [13]. SWAM is a pattern-oriented tool that performs a variety of tasks from segmentation to semantic search. GPOW2 improved the time performance of SWAM by more than 90% while maintaining the high accuracy of the engine.

The test accuracies of GPOW2 predictions are 99.24% for Arabic morphological patterns and 98.80% for POS tags respectively. These results beat the SWAM Gold Standard results obtained for the same data set and are much higher than predictions produced by ALShgar (2012). More work is needed to extend and improve the emerging GPOW2 model in different directions and apply it to other types of tasks.

Acknowledgement

This work has been funded by EIAS Data Science & Blockchain Lab, Prince Sultan University. The authors are very grateful to the Lab and Prince Sultan University

References

[1] Alashqar A. A comparative study on Arabic POS tagging using Quran Corpus. In: Proc. Int. Conf. Informatics and Systems (INFOS) (pp. 29–33), 2012.
[2] Altowayan AA, Tao L. Word embeddings for arabic sentiment analysis. In: Int. Conf. Big Data (Big Data). IEEE, 2016.
[3] Bengio Y, Ducharme R, Vincent P, Jauvin C. A neural probabilistic language model, J Machine Learn Res, vol. 3, no. Feb, pp. 1137–1155, 2003.
[4] Blei DM, Andrew, Ng Y, Jordan MI. Latent Dirichlet Allocation. J Mach Learn Res 2003;3(4–5):993–1022. doi: https://doi.org/10.1162/jmlr.2003.3.4-5.993.
[5] Buckwalter T. Buckwalter Arabic Morphological Analyzer Version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0, 2012.
[6] Collobert J, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P. Natural language processing (almost) from scratch. J Mach Learn Res 2011;12:2493–537.
[7] Deerwester S. et al. (1988). Improving Information Retrieval with Latent Semantic Indexing. In: Proc. 51st Annual Meeting of the American Society for Information Science 25, p. 36–40.
[8] Delvin J, Chang M, Lee K, Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, arXiv:1810.04805 [cs.CL], 2018.
[9] Dhaou A, Xiong S, Zhou J, Haddoud MH. Word Embeddings and Convolutional Neural Network for Arabic Sentiment Classification. In Proc. COLING 2016, 26th Int. Conf. Computational Linguistics: Technical Papers (pp. 2418–2427), 2016.
[10] Diab, M. Towards an optimal POS tag set for Modern Standard Arabic processing. In: Proc. Recent Advances in Natural Language Processing (RANLP). pp. 91–96, 2007.
[11] Dukes K, Habash N. (2010a). Morphological Annotation of Quranic Arabic. In: Proc. Language Resources and Evaluation Conference (LREC) (25302536), Valletta, Malta.
[12] Dukes K, Atwell E, Sharaf A. Syntactic Annotation Guidelines for the Quranic Arabic Dependency Treebank. In: Proc. Language Resources and Evaluation Conference (LREC), Valletta, Malta, pp. 1822–1827, 2010b.
[13] ELAffendi M, Abuhaimed I. SWAM Arabic Linguistic Toolkit (SALT): A Novel Hybrid Word Embedding Neuro Tool for Performing Downstream NLP Tasks, unpublished, 2019.
[14] ELAffendi MAKS, Alrajhi KS. Text Encoding for Deep Learning Neural Networks: A Reversible Base 64 (Tetrasexagesimal) Integer Transformation (RIT64) Alternative to One Hot Encoding with Applications to Arabic Morphology, In: 6th Int Conf. Digital Communications, Networking, and Wireless Communication, Lebanese University, Beirut, Lebanon, pp. 70–74, 2018.
[15] ELAffendi MA. A Suggested Framework for Arabic Morphological Analysis: A Sliding Window Asymmetric Matching Algorithm and its Implication, Egypt Informatics J, Cairo University, vol. 9, no. 1, June, 2008.
[16] ELAffendi MA. The Generative Power of Arabic Morphology and Implications: A Case for Pattern Orientation in Arabic Corpus Annotation and a Proposed Pattern Ontology. In: Alenezi M, Qureshi B (eds.), 5th Int Symp. Data Mining Applications. Advances in Intelligent Systems and Computing, vol. 753. Springer, Cham, 2018.
[17] ELAffendi MA, Altayeb M. The SWAM Arabic Morphological Tagger: Multilevel Tagging and Diacritization, Using Lexicon Driven Morphotactics and Viterbi, ICAI'14" 2014 Int. Conf. Artificial Intelligence, Las Vegas, Nevada, USA, 2014.
[18] Habash N, Rambow O, Roth R. MADA+TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In: Choukri, K. and Maegaard, B. (eds.), Proc. 2nd Int. Conf. Arabic Language Resources and Tools. The MEDAR Consortium, in Proc. LREC, Reykjavik, Iceland, 2009.
[19] Howard J, Rud.er S. Universal Language Model Fine-tuning for Text Classification. 1801.06146v5 [cs.CL] 23, 2018.
[20] Joulin A, Grave AE, Bojanowski E, Mikolov T. Bag of Tricks for Efficient Text classification. arXiv: 1607.01759v3 [cs.CL], 2016.
[21] Khoja S, Garside R, Knowles G. A tag set for the morphosyntactic tagging of Arabic. Comput. Dep. Lanc. Univ; 2001.
[22] Kusner MJ, Sun Y, Kolkin KI, Weinberger KQ. From Word Embeddings to Document Distances. in Proc. 32nd Int. Conf. Machine Learning, Lille, France, JMLR: W&CP vol. 37, 2015.
[23] Kythe Prem K, Kythe Dave K. Algebraic and stochastic coding theory. CRC Press; 2017.
[24] Mijangos V, Sierra G, Herrera A. Word embedding model for sentence similarity. Res Comput Sci 2016;117:63–74.

[25] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space, arXiv preprint arXiv: 1301.3781, 2013.
[26] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality, Adv Neural Inf Proces System, pp. 3111–3119, 2013.
[27] Mikolov T, Le QV, Sutskever I. Exploiting Similarities among Languages for Machine Translation", rXiv: 1309.4168v1, 2013.
[28] Mitra B, Nalisnick E, Craswell N, Caruana R. A Dual Embedding Space Model for Document Ranking. arXiv preprint arXiv:1602.01137 February 4, 2016.
[29] Mohamed E, Kübler S. Arabic part of speech tagging. In: Proceedings of LREC, Valetta, Malta, 2010.
[30] Nalisnick E, Mitra B, Craswell R. Improving Document Ranking with Dual Word Embeddings. WWW'16 WWW – World Wide Web Consortium (W3C), 2016.
[31] Pasha A, Al-Badrashiny, Kholy AE, Eskander R, Diab M, Habash N, Pooleery M, Rambow O, Roth R. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In: Proc. Language Resources & Evaluation Conf, 2014.
[32] Petersy ME, Neumanny M, Mohit Iyyery M, Matt Gardnery M, Clark C, Lee K, Zettlemoyer L. Deep contextualized word representations, arXiv: 1802.05365v2, 2018.
[33] Rajan D, Chen T, Serpedin E. Mathematical foundations for signal processing, communications, and networking. CRC Press; 2017.
[34] Dumais ST. Latent Semantic Analysis. Annual Review of Information Science and Technology, vol. 38, pp. 188–230, 2005. doi: 10.1002/aris.1440380105.
[35] Zeroual I, Lakhouaja A, Belahbib R. Towards a standard Part of Speech tag set for the Arabic language. J King Saud Univ – Comput Information Sci 2017;29(2):171–8.
[36] Zhang M, Yu N, Fu G. A Simple and Effective Neural Model for Joint Word Segmentation and POS Tagging, IEEE/ACM Trans. Audio, Speech, Language Process., vol. 26, no. 9, 2018.