Deep Learning Framework with Confused Sub-Set Resolution Architecture for Automatic Arabic Diacritization

Mohsen A. A. Rashwan, Ahmad A. Al Sallab, Hazem M. Raafat, and Ahmed Rafea

***Abstract*—The Arabic language belongs to a group of languages that require diacritization over their characters. Modern Standard Arabic (MSA) transcripts omit the diacritics, which are essential for many machine learning tasks like Text-To-Speech (TTS) systems. In this work Arabic diacritics restoration is tackled under a deep learning framework that includes the Confused Sub-set Resolution (CSR) method to improve the classification accuracy, in addition to an Arabic Part-of-Speech (PoS) tagging framework using deep neural nets. Special focus is given to syntactic diacritization, which still suffers low accuracy as indicated in prior works. Evaluation is done versus state-of-the-art systems reported in literature, with quite challenging datasets collected from different domains. Standard datasets like the LDC Arabic Tree Bank are used in addition to custom ones we have made available online to allow other researchers to replicate these results. Results show significant improvement of the proposed techniques over other approaches, reducing the syntactic classification error to 9.9% and morphological classification error to 3% compared to 12.7% and 3.8% of the best reported results in literature, improving the error by 22% over the best reported systems.**

***Index Terms*—Arabic diacritization, classifier design, deep networks, part-of-speech (PoS) tagging.**

I. Introduction

ARABIC is a wide spread language spoken by over 350 million people on the planet. The Arabic alphabet and vocabulary are very rich, with the same word morphology being a candidate of different meanings and pronunciations. For example the word رمع might bear the meaning of the person name "Omar" عَمْر or the meaning of "age" عُمْر What distinguishes them is the diacritization signs assigned to each character of the word.

Manuscript received July 14, 2014; revised November 21, 2014; accepted January 12, 2015. Date of current version February 26, 2015. The guest editor coordinating the review of this manuscript and approving it for publication was Prof. Isabel Trancoso.

M. A. A. Rashwan is with the Engineering Company for the Development of Computer Systems (RDI), Giza 12613, Egypt, and also with the Department of Electronics and Electrical Communications, Faculty of Engineering, Cairo University, Giza 00202, Egypt (e-mail: mohsen_rashwan@rdi-eg.com).

A. Al Sallab is with Valeo Interbranch Automotive Software, Giza, Egypt, and also with the Department of Electronics and Electrical Communications, Faculty of Engineering, Cairo University, Giza 12613, Egypt (e-mail: ahmad.el-sallab@valeo.com).

H. M. Raafat is with the Computer Science Department, Kuwait University, Safat 13060, Kuwait (e-mail: hazem@cs.ku.edu.kw).

A. Rafea is with the Department of Computer Science, American University in Cairo (AUC), Cairo 11835, Egypt (e-mail: rafea@aucegypt.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2015.2395255

Diacritics are short vowel marks added on the character to reflect its correct pronunciation, according to grammatical, syntactical and morphological rules of the language.

Historically, diacritics were not included in Classical Arabic (CA). Even the holy Quran was originally written with neither diacritics nor pointing system. Later, Muawiyah of the Umayyad dynasty asked linguists and scientists in Basra to introduce a diacritization system, which was developed first by Abu al-Aswad al-Du'ali, then tailored and improved by Al-Farahidi to develop into the system known today.

Nowadays, Modern Standard Arabic (MSA) transcripts are written without diacritics, leaving the reader to restore them from the context and knowledge. Diacritics restoration is not an easy task even for knowledgeable, native Arabic speakers. On the other hand, there are many machine learning tasks, like Text-To-Speech (TTS), translation, spelling correction, word sense disambiguation, etc., that require diacritizing the script as a pre-processing step before applying the core algorithms.

In its basic form, the problem can be reduced to a pattern classification problem, with seven diacritics classes being the targets. In addition, the diacritics classification can be divided into syntactical diacritization, caring about case-ending and morphological diacritization, caring about the rest of the word diacritics. Currently, the morphological part of the problem is almost solved, leaving a marginal error of around 3-4%. On the other hand, syntactic diacritization errors are still high, hitting a floor that is claimed to be asymptotic and cannot be further reduced ([1], [2]). For this reason, we focus our effort on reducing the error below the 12.5% error obtained in [1] and [2].

Recently, a significant advancement in the area of deep learning has been witnessed, with the development of a generative model, Deep Belief Nets (DBN), with a fast algorithm for inference of the model parameters. Deep Neural Networks (DNN) shall be the basic machine learning classifier used in this work, employing the latest results reached in the deep learning field. An efficient feature vector is designed under the umbrella of deep learning to distinguish different word diacritics. Features that are tested in the current work are: PoS, morphological quadruple of lexemes, last character and word identity. In addition, context features are essential to the diacritization problem. Context features include the previous word's features, as well as the previous word's diacritics.

Part-of-Speech (PoS) features are critical to syntactic diacritization, which is the focus of this work. For some datasets PoS tags are manually annotated by professional linguists, while for the real case and most datasets, they are not available. For this

reason, standalone PoS taggers are built under the deep learning framework, which can reused in Arabic PoS tagging systems, needed for many other applications, and not only for Arabic diacritization.

The deep learning model often hits a performance barrier which cannot be crossed. Hence, error analysis and diagnosis is run on the confusion matrix results, leading to the Confused Sub-set Resolution (CSR) method to train sub-classifiers to resolve the identified confusions and automatically generate a deep network-of-networks composed of the main classifier and the sub-classifiers working together to offer around 2% error enhancement.

Evaluation of the proposed techniques is done on two datasets; the first is a custom one collected from many different sources, which is available online at [10]. Manually extracted PoS and morphological quadruples are available for only a part of this dataset. The PoS tags of this part of the dataset were used to build the DNN PoS taggers to tag the rest of the dataset. The corresponding test set, which is available online at [11], was collected from different sources than training ones and is quite challenging. The second dataset is the standard LDC Arabic Tree Bank dataset [12] used to bench mark the system against state-of-the art systems in Arabic diacritization area.

The rest of the paper is organized as follows: first the related works in literature are surveyed, followed by a formulation of the CSR method. The next section is dedicated to describing the features used in the system, and how they are encoded and represented in the feature vector followed by the details of building the DNN PoS tagger for Arabic. The datasets used for evaluation are then described. The next section describes the system evaluation experiments. Experimental results include an error analysis study of the effect of each feature and method on the system performance, in addition to benchmarking against state-of-the art systems in literature, evaluated on standard datasets. Finally, the paper is concluded with the main results and conclusion.

II. Related Work

There have been many attempts to approach the Arabic diacritization problem by different techniques. Focus will be around three works strongly related to what is proposed here, and having the best results in literature. Zitouni *et al.* (2006) [3] apply Maximum Entropy classification to the problem taking the advantage of the MaxEnt framework to combine different features together, like lexical, segment-based, and PoS features. Segmentation involves getting the prefix, suffix, and stem of the word. PoS features are also generated under the MaxEnt framework. Habash and Rambow (2007) [4] perform Morphological Analysis and Disambiguation of Arabic (MADA) system, and then apply SVM classification. Last, Rashwan *et al.* (2010 and 2011) [1], [2] and [2] propose a hybrid approach composed of two stages: first, maximum marginal probability via A* lattice search and n-grams probability estimation. When full-form words are OOV, the system switches to the second mode which factorizes each Arabic word into all its possible morphological constituents, then uses also the same techniques used by the first mode to get the most likely sequence of morphemes, hence the most likely diacritization. The latter system ([1], [2]) shall be our baseline, since it gives the best results in literature so far, and the dataset used to evaluate it is available at our hand, and hence fair comparison is possible. Also, comparison to the three systems is made on the LDC Arabic Tree Bank data set.

TABLE I
Arabic Diacritics Classes

Diacritics form on Arabic letter ب	Class name	Pronunciation
بَ	Fatha فتحة	/a/
بُ	Damma ضمة	/u/
بِ	Kasra كسرة	/i/
بً	Fathten فتحتين	/an/
بٌ	Dammeten ضمتين	/un/
بٍ	Kasreten كسرتين	/in/
بْ	Sukun سكون	No vowel
بّ	Shadda شدّة	Double consonant

III. Deep Learning Framework

The Arabic diacritics restoration task can be formulated as pattern classification problem. The target classes shall be the diacritics themselves, described in Table I. The input is the raw MSA transcript. The task is to classify the input based on well-designed feature vector and restore the original diacritics of the raw text. The output shall be the full diacritized text. All these diacritics can exist on case-ending character, while Fathten, Dammeten and Kasreten can never occur on non-ending character of the word root.

The machine learning classifier tool chosen in this paper is the Deep Neural Network (DNN), under the framework of learning deep architecture proposed by Hinton *et al.* (2006) [5] and [6]. The raw text is presented to the classifier, and a group of sub-nets work to extract the desired features, like PoS tags. The network architecture is shown in Fig. 1. Each sub-net is trained to extract a certain kind of features, and the obtained feature vectors are concatenated together to form the input that is represented to the classifier network. In fact the training of features extraction nets is guided by certain desired features, like PoS tags. This enables building a standalone system that operates on the raw text only.

IV. Confused Sub-set Resolution Method

The Confused Sub-Classes Resolution (CSR) is based on confusion matrix analysis and the method in [7]. The output of this analysis shall be a network architecture composed of the original classifier operating with sub-classifiers to resolve confusions that were identified through confusion matrix analysis.

The method starts with training a global classifier, then evaluating its performance. To enhance its accuracy, the sources of errors are analyzed by building the confusion matrix for the training set. The position of the off diagonal element identifies the pair of classes that are confused together.

The confused classes are grouped together in confused sub-sets. Each pair sharing a class target are grouped together forming new entry in the set, and so on until no more grouping

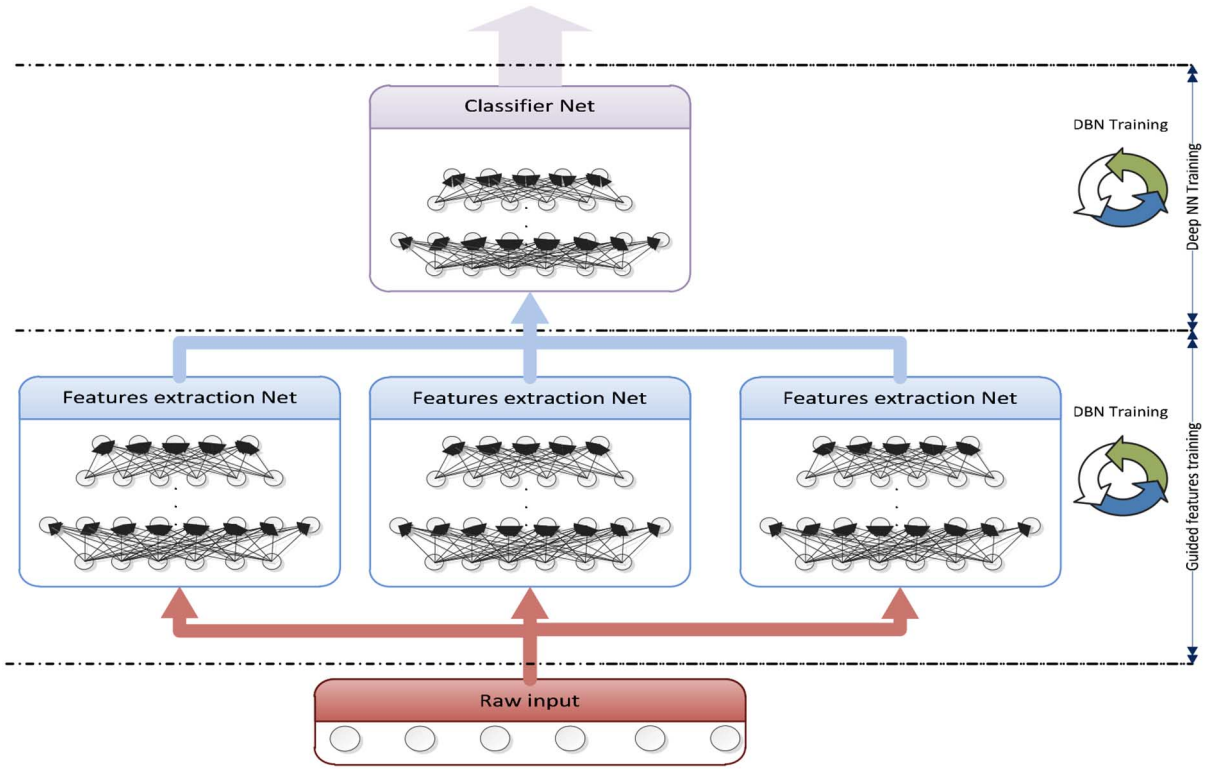

Fig. 1. Deep network framework.

is possible. For each member group of the final set, a sub-classifier is trained, only using the training cases that hold the label of the sub-class members of the group.

Once the groups of sub-classifiers are available, a deep network model is formed out of the global classifier and the sub-classifiers, such that the first layer is formed of the global classifier, followed by the second layer of the sub-classifiers groups having the maximum number of members, followed by a deeper layer of fewer members groups, and so on. The training or test cases are fed to this architecture so that each case is fed to the global classifier, and then based on its decision the example is routed to the proper sub-classifiers to resolve any possible confusion. The same procedure is repeated in each layer till the last layer resolving conflict between pair of classes.

A. Algorithm

The flow chart of the CSR method is shown in Fig. 2. The following steps describe the algorithm:

1. Train a basic global classifier in DNN framework and obtain the confusion matrix C on the training set
2. Identify the confusion domains $D = \{D^i\}$ that have confusions more than a threshold δ, which is a parameter of the algorithm obtained from confusion matrix analysis. It could be set by sorting the off-diagonal elements in order of magnitude, and selecting the threshold to be in the middle of the largest gap in magnitude between successive elements, similar to the "eigengap" heuristic used in spectral clustering.
3. Train sub-classifiers for each confusion domain D^i.
4. Determine the architecture of the model having N^{nm} sub-classifiers. The superscript n denote the index in the layer, while m denotes the layer depth in which this domain is resolved. When a sub classifier is a complete subset of another one, it is placed in a deeper layer of the architecture. In this case, the m superscript is incremented to denote extra depth in the model.

B. Network Model

1- Arabic diacritics classes contain high confusion domains, for example the "Kasra" and "Kasreten" classes are expected to be highly confused. In an attempt to resolve this confusion, CSR algorithm is applied. The first step is to generate the confusion matrix. Table II shows the confusion results for DNN classifier. Fatha, Damma, Kasra: $D^{11} = \{4, 5, 6\} \rightarrow \mathrm{N}^{11}$ with threshold $\delta = 2000$
2- Fathten, Dammeten, Kasreten: $D^{21} = \{1, 2, 3\} \rightarrow \mathrm{N}^{21}$
3- Kasra, Kasreten: $D^{12} = \{3, 6\} \rightarrow \mathrm{N}^{12}$

Each domain has its own N^{nm} classifier to resolve its confusions. The final model shall be as shown in Fig. 3. The identified confusions go exactly with the Arabic grammatical rules which directly influence the syntactic diacritization task. An interesting property arises here which is the ability to inject sub-class specific features to further resolve confusions. For example the sub-classifier N^{12} resolving Kasra/Kasreten is highly impacted if the concerned word has identification pre-fix ﺎﻟ which is a feature definitive enough to resolve the confusion, so this specific feature can be given higher weight or specially handled for this sub-classifier. However, this is left to future works on CSR since the scope of the current work does not include handcrafted features with task-specific knowledge.

TABLE II
CONFUSION MATRIX RESULTS FOR DNN CLASSIFIER ON SYNTACTIC DIACRITIZATION

	Fathten	Dammeten	Kasreten	Fatha	Damma	Kasra	Shadda	Sukkun
Fathten	4762	2179	2455	336	389	197	0	120
Dammeten	2647	6976	2720	660	1144	408	0	231
Kasreten	4560	3378	32588	801	303	**4868**	0	951
Fatha	438	475	1458	92755	11671	8340	0	1980
Damma	262	727	579	5858	72994	14995	0	952
Kasra	59	184	**3275**	2682	3657	220357	0	1970
Shadda	2	78	86	51	75	0	416	4
Sukkun	3	128	271	1150	630	1565	0	73983

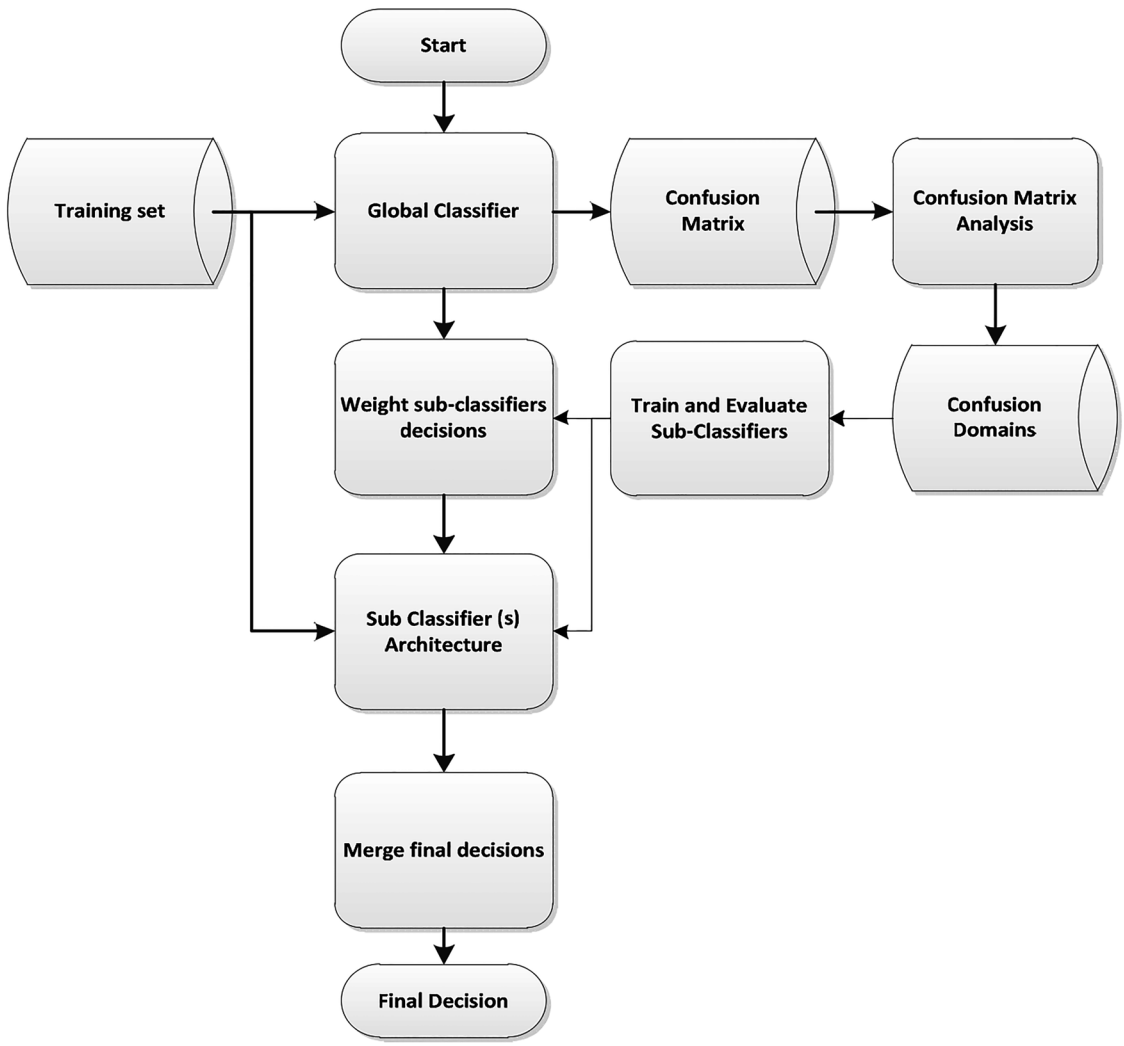

Fig. 2. CSR algorithm flow chart.

C. Probabilistic Interpretation

Multi-class classification problems are concerned with minimizing an error function $J(\gamma, S_{train})$ with respect to the classifier parameters γ on the training set $S_{train} = \{(x^i, y^i) : i = 1, .., N\}$ made of N examples, each represented with feature vector x^i and label y^i. The minimization of the error is equivalent to maximizing the discrimination probability, or equivalently, maximizing the classification accuracy:

$$\gamma^* = \arg\min_{\gamma} J(\gamma, S_{train}) \equiv \arg\max_{\gamma} p(Y = y^i | X = x^i)$$

The random variable $X \in \Re^{1\times X}$ is the feature vector, $Y \in \{Y^1\ Y^2 \ldots Y^M\}$ is the class label and M is the number of class labels. The accuracy of classification is linked to how efficient the discrimination probability models the real data.

We propose to factorize the class labels set into clusters or sub-classes mostly confused together: $Y = C^1 \cup C^2 \cup \ldots \cup C^L$ where L is the number of sub-classes, so that this probability is as follows:

$$p(Y = y^i | X = x^i) = \sum_{j=1}^{L} p(c = C^j | X = x^i) \times p(Y = y^i | X = x^i, c = C^j)$$

This factorization is similar to the approach in [8] but with a different factorized random variable. While we factorize the

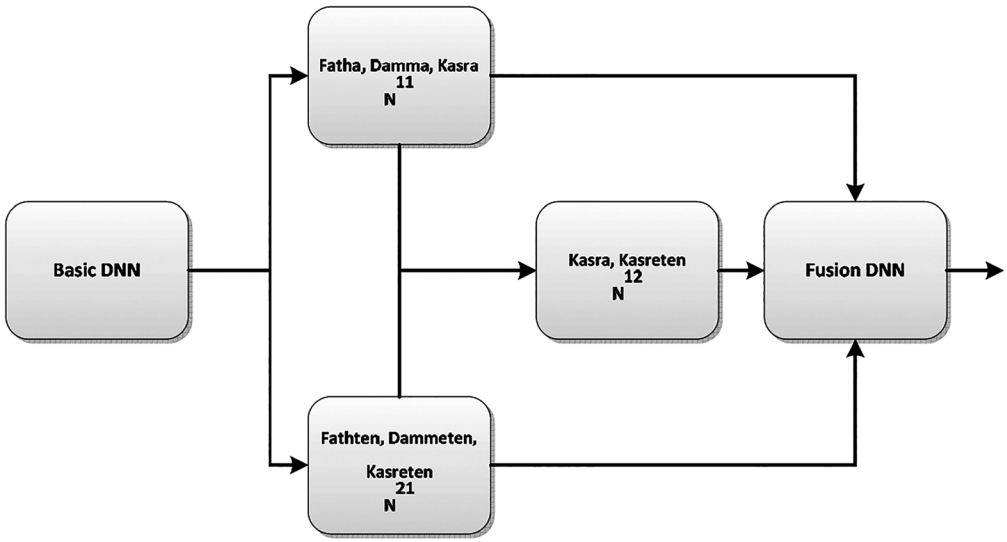

Fig. 3. CSR model for syntactic Arabic diacritization task.

class labels, the method in [8] factorizes the word input vocabulary or the language model for quite a different objective. Moreover, the clusters are identified in our work based on confusion matrix analysis.

In the CSR framework, each sub-class is assigned a sub-classifier to discriminate its members, which is meant to model $p(Y = y^i|X = x^i, c = C^j)$. Also, each sub-class is assigned a domain classifier to trigger it, which is meant to model $p(c = C^j|X = x^i)$.

The success of the approach is strongly determined by finding a set of sub-classifiers such that their accuracy in the domain of sub-classes they discriminate is higher than that of the basic classifier modeling $p(Y = y^i|X = x^i)$ directly. The gain in accuracy reflects the correction capability of the sub-classifier. It is expected that the confusion matrix after applying CSR is better shaped with less sum of off-diagonals than the basic case. Further iterations of the algorithm are possible to obtain a more improved confusion matrix having maximum diagonal elements.

Also, the detection capability $p(c = C^j|X = x^i)$ which triggers the sub-classifiers should be improved or at least not degraded. If we count on the basic classifier decision to trigger the sub-classifiers, then we have the same $p(c = C^j|X = x^i)$ as the basic model, which means that the improvement is governed by the improvement in $p(Y = y^i|X = x^i, c = C^j)$.

The above probabilistic interpretation of the method enables to get rid completely of the fusion DNN in Fig. 3, which functions as a selector trained to take the output from the most appropriate classifier of the network. The factor $p(c = C^j|X = x^i)$ replaces this DNN, triggering only the good classifier and dumping the others. This simplifies the model than and yields similar results.

V. Features

The input to text processing tasks is a transcript or document containing raw text. For Arabic diacritization task specifically, a set of features have proved good performance in literature, such as morphological lexemes, PoS, word identity,…etc see [1], [2], [3] and [4]. In this section the features employed in our feature vector are described.

Arabic morphological model assumes the canonical structure uniquely representing any given Arabic word "w" to be a quadruple of lexemes (or morphemes) so that w $\rightarrow$ q = (t: p, r, f, s) where p is prefix code, r is root code, f is pattern (or form) code, and s is suffix code. Morphological features are normally more effective in morphological than syntactic diacritization, since they are concerned with the word morpheme or building.

Syntactic diacritization is about adding diacritics on the last character of the word. Arabic language prohibits some diacritics from being placed over some characters. For example fatha on"ى" is phonetically forbidden. Also, it favors some diacritics over some character like fatheten on "ا". A rule based system would have set a rule for that, however, in DNN framework, the system is left to learn such rules. Hence, the last character identity is an effective feature for syntactic diacritization task.

The raw word identity is another type of possible features. The system in [1] uses this feature. There are two possibilities of encoding such feature, the first would be to use a raw index representing the word index from a vocabulary vector built from training set. However, this could lead to many out of vocabulary (OOV) cases, in addition to a long vector. On the other hand, a word can be encoded as sequence of the identities of its composing characters, which is more efficient under the DNN framework to avoid OOV, because even if a word is not encountered during training, at least a similar one with a character less or more was encountered during training, generating nearly similar activation of the stochastic binary units and leading to similar result as the original word. The same exact word need not be present during training phase, instead only a similar word is enough so that the word is not considered OOV. This is a direct result of encoding words as sequence of their constituting characters.

Considering the feature vector of the preceding and/or succeeding words or characters can improve significantly the classification accuracy. This is what we refer to as context features. Context features are essential to syntactic and morphological

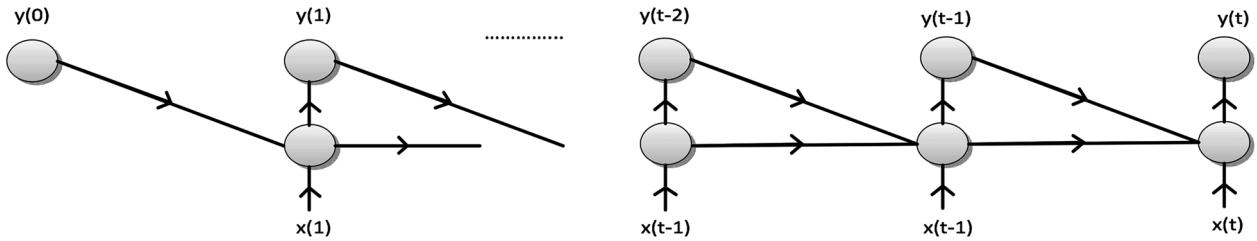

Fig. 4. Class context features N = 1, M = 0.

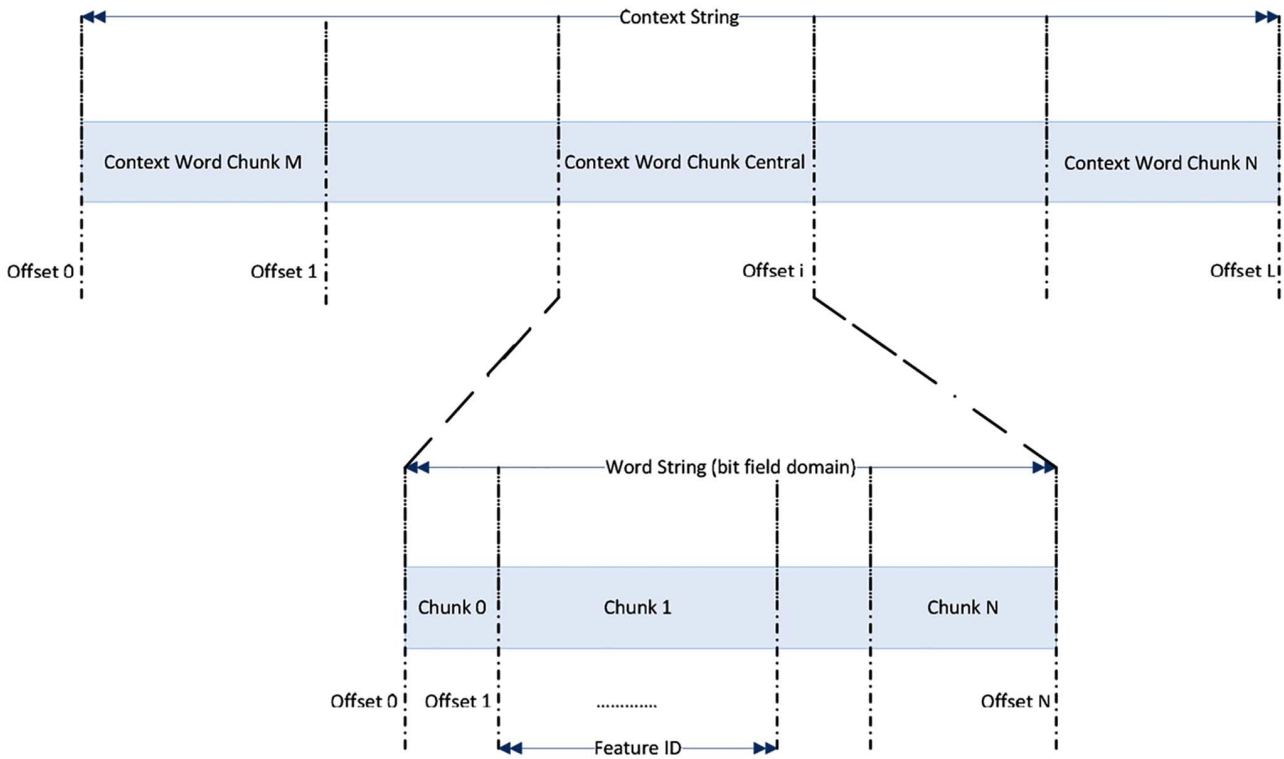

Fig. 5. Context and word features sparse representation.

diacritization tasks. For morphological diacritization context is just the surrounding characters, while for syntactic diacritization context is represented by the surrounding words. We denote the depth of the preceding context by N and the succeeding context elements by M (see Fig. 6).

The context does not only include the feature vectors of inputs, it can also include the context of class labels. For example, the decision of the classifier for the previous diacritic can be re-considered as an input feature for the current diacritic classification. This results in something like an auto-regressive model, where the previous decisions affect the next one recursively. We refer to the input features context as $X_{context}$ and to the class labels context as $Y_{context}$.

In the model described in Fig. 4 only the previous label is considered for simplicity (N = 1, M = 0). The class labels context $Y_{context}$ is appended to the feature vector just the same as the input context, so both are embedded in the same vector. In this case, the previous class label is obtained as the output of the classifier for the previous input $y(t-1)$, resulting in a recursive, or auto regressive model. The DNN network is then left to learn how to discriminate current class label $y(t)$ based on the current input $x(t)$, the previous input $x(t-1)$ and the previous output $y(t-1)$ all merged in one feature vector.

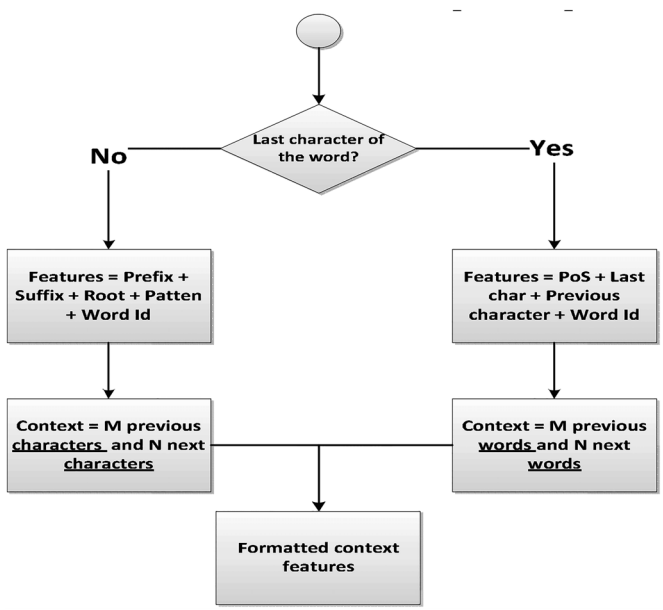

Fig. 6. Context features in case of last character and rest of word diacritization.

To complete the picture all features used in the system are listed in Table III. Each feature is represented by an identifier

TABLE III
EMPLOYED FEATURES AND THEIR DYNAMIC RANGE

Feature	Range($B_{feature}$)	Offset ($o_{feature}$)
Prefix	[1 : 255]	0
Root	[1 : 4446]	255
Form	[1 : 973]	4701
Suffix	[1 : 5387]	5674
PoS tag	[1 : 24]	11061
Word identity	[1 : 28]x[25]	11087
Last character	[1 : 28]	11787
Previous diacritic	[1:7]	11815

TABLE IV
SAMPLE ARABIC POS TAGS VECTORS

Sample word	Arabic PoS tags vector
مَوَاضِيع	[Null Prefix, Noun, No SARF, Plural, Null Suffix] [لا جمع، الصرف، من ممنوع اسم، سابقة، لا لاحقة]
ٱلْكِتَابَات	[Definitive, Noun, Plural, Feminine] [مؤنَّث، جمع، اسم، ال لتعريف]

(ID) that can take integer value from a certain range; hence each feature ID has its own dynamic range depending on the number of values it can take. For example, the "Prefix" feature can be one of 255 prefixes types, so its ID value can take any value from 1 to 255.

Merging all features together in one vector can be done based on their raw indices as in Fig. 5, but in this case the features with higher dynamic range will dominate and bias the classification decision. Hence, some sort of normalization is required. In our implementation, we choose a special sort of normalization technique based on assigning a vector for each feature with its length equal the dynamic range of the feature. Each element of such vector is either 0 or 1 depending on the value the feature takes. This technique is similar to one-hot spot encoding, but to overcome the problem of missing the similarity between semantically similar words, we encode the context word features as well. In some sense, encoding the context in the same vector achieves the same goal of clustering using LSA or LDA, or even neural word embeddings [9], because it is based on the same concept; similar words tend to appear in the same context.

VI. POS TAGGING

Part-of-Speech tags are essential features to discriminate syntactic diacritics cases, where syntactic diacritics restoration is strongly related to grammatically parsing and analyzing the sentence into its syntactic language units or PoS. There are many models for Arabic PoS tags. In this work we adopt the one in [2] (example in Table IV), which sets 62 context-free atomic units to represent all possible Arabic language PoS tags. A very rich dataset of Arabic words, extracted from different sources, is used to train the system (see VIII TRN_DB_I and TST_DB). PoS tags are manually annotated for this dataset by expert Arabic linguists. The PoS tags for each word are presented in the form of a PoS tag vector, with a word assigned

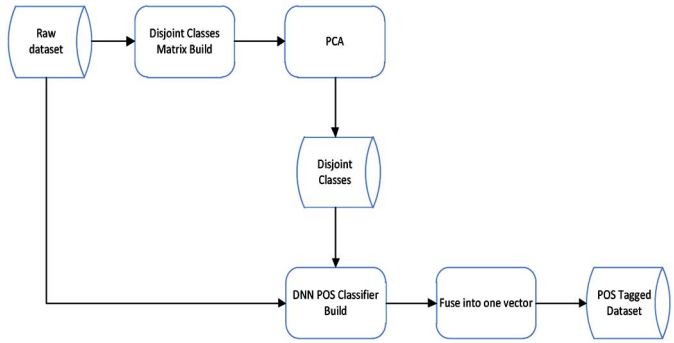

Fig. 7. Block diagram of disjoint classifiers PoS tagging method.

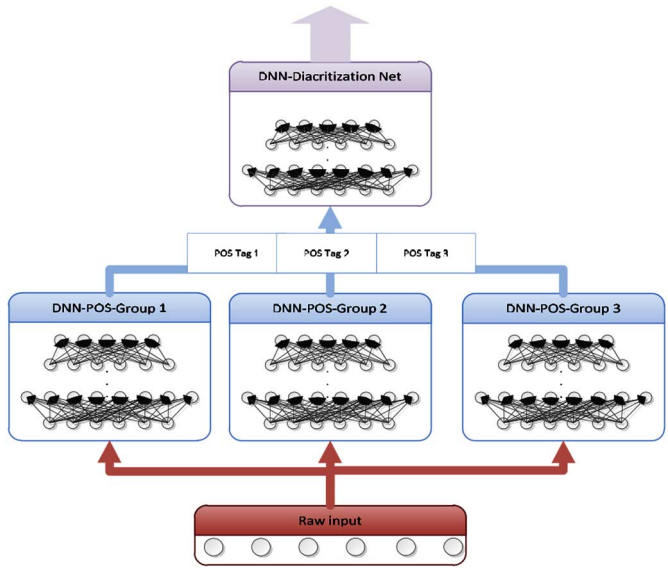

Fig. 8. Deep learning framework for DNN-PoS-diacritization.

many PoS tags at once (PoS vector), see Table II in [1], which is partially replicated here for clarity:

However, to be able to use this dataset under the deep learning framework, a word must be assigned one and only one tag representing its target class. Hence, a method was developed to analyze the PoS vectors into single tag per word. In fact, each tag given in the vector can be viewed as separate feature.

The length of those features is 62 different tags. By analyzing the dataset it was found that up to 11 different tags could be assigned to one word at a time.

To be able to train a PoS classifier, it is important to identify the different tag groups that compose the PoS tags in the dataset. Some tags always occur together, offering no extra information, while others never occur together, indicating that they are members of the same feature set like, for example, "Noun" and "Verb", which normally do not occur together, and hence belong to the same feature group. A feature group is a set of PoS tags that do not occur together. To get rid of redundancy PCA can be run to reduce the members of each feature set. We refer to the independent feature sets as "Disjoint Classes".

Fig. 7 describes the process. First the raw dataset is parsed to form a matrix of POS tag co-occurrences. The matrix is composed of 62×62 entries, with an entry per PoS tag pair. For each word encountered, the joint cell of the matrix is incremented; to indicate how many times the two tags occur together. After the

TABLE V
REDUCED POS TAGS GROUPS

Group	PoS Tags
Group 1	(Future)
	(ParticleNAASSIB)
	(Present)
	(Imperative)
	(Noun)
	(PrepPronComp)
	(JAAZIMA)
	(CondJAAZIMA)
	(CondNotJAAZIMA)
	(LAA)
	(Except)
	(ParticleNAASIKH)
	(VerbNAASIKH)
	(Past)
Group 2	(MAJZ)
	(MARF)
	(MANSS)
	(MANS_MAJZ)
	(MANSS_MAGR)
	(MAGR)
Group 3	(NullSuffix)
	(PossessPro)
	(ObjPossPro)
	(ObjPro)

whole dataset is parsed, tags that never occur together are members of disjoint groups of feature sets. To eliminate redundancy, PCA is run to remove entries that always occur together offering no extra information. After this step, three group where identified in Table V:

Hence, the 62 tags were reduced to 24 non-redundant and non-overlapping tags, divided into three groups that can occur simultaneously in one PoS tag vector. For each group, a separate DNN classifier is trained. The final PoS tag is taken as the fusion of the three classifiers. This feature vector is the input to the next stage DNN diacritization classifier according to the deep architecture presented in III Fig. 1 and Fig. 8. The input to the DNN-PoS Tagger is just the raw text, with each word encoded in its sequence of characters, in addition to the context words surrounding the word to be tagged. The output is the PoS tag assigned to this word. The feature vector is formed as $x = [y^1_{POS} y^2_{POS} y^3_{POS}]$

VII. OVERALL SYSTEM ARCHITECTURE

Throughout the previous sections, the main building blocks of the proposed system are described. In this section the overall system is presented in a coherent way. Fig. 9 describes the architecture of the proposed system, putting the pieces of the puzzle together to form the big picture.

The raw text input is fed to the system word by word. According to the configured context depth, a number of succeeding and preceding words are stored in a context memory. In our system the context is empirically taken as three preceding words and one succeeding word (N = 3, M = 1), which is found to give the best accuracy results. This context will serve later to format the context feature vector. If the word is the first or last one in a sentence, the preceding or succeeding context is zero padded. Word context serves in case of syntactic diacritization, while for the morphological case the character context is also

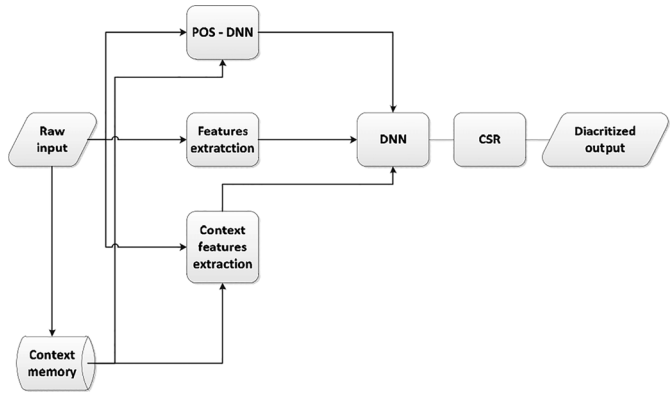

Fig. 9. Overall Arabic diacritization system.

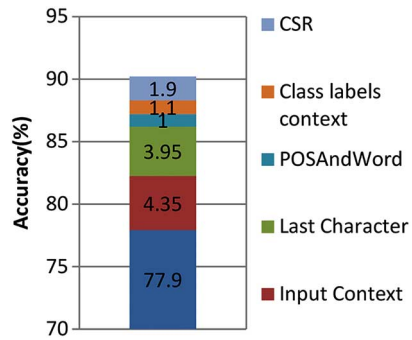

Fig. 10. Contributions to the overall accuracy.

needed, which is directly present in the character sequence of the single input word itself. Context encoding is described in details in V.

The feature extraction procedure depends on the feature itself. For PoS tags, a special DNN is trained for that purpose, which also makes use of the context of the word. For other features, like the sequence of characters forming the word, the last character of the word and the morphological quadruples are directly extracted from the single word.

The framework in Fig. 1 is employed. A three-layer network architecture is used for each feature extraction subnet or classification network. For the classification network a 20-20-20 architecture was used, while for PoS-tagging networks a 60-60-60 architecture is used. The network architecture is determined empirically. By experiments it was found that the best architecture is the symmetric one, with the same width for all layers. The best width is found to be the same as the average number of ones in the training set feature vectors. To clarify, in our system, the feature vectors are sparse bit fields with few positions set to one and the rest of the vector are zeros. To get the best network width, we count the number of ones in each vector of the training set and average this count. The resulting average is found to be the best setting of the net layers widths.

The neural network training undergoes DBN pre training as in [5] and [6] for 20 epochs per layer, with batch size of 1000 examples each without mini batches. Momentum is used initially with 0.5 for the first 5 epochs and then raised to 0.9 for the next epochs. The discriminative fine tuning is performed using conjugate gradient minimization for 30 epochs. For the first 6 epochs, only the upper layer is adjusted, then the rest of the layers are trained for the next epochs.

TABLE VI
EFFECT OF CSR

Dataset	Accuracy with CSR (%)	Accuracy without CSR (%)
TRN_DB_I – Ready PoS	90.2	88.2
TRN_DB_I – Raw text	88.2	86.2

TABLE VII
EFFECT OF CLASS LABELS CONTEXT ON SYNTACTIC DIACRITIZATION

Dataset	Accuracy with class labels context (%)	Accuracy without class labels context (%)
TRN_DB_I – Ready PoS	88.2	87.2
TRN_DB_I – Raw text	86.7	85.1
TRN_DB_I + TRN_DB_II / TST_DB	86.2	84.3

Once the features are ready for a given word, they are fed to the DNN classifier. The resulting confusion matrix from the training phase is then fed to the CSR method to generate the tree structure that improves the system accuracy. During testing, the raw input features are fed to the DNN classifier to obtain an initial guess of the target diacritic. This guess is then improved in the next CSR stage to obtain the final diacritic decision.

VIII. DATASETS

The following datasets are used in our experiments:

- ***TRN_DB_I:*** This is a 750,000 word dataset, collected from different sources and manually annotated by expert linguists with every word PoS and Morphological quadruples. For more information refer to [1].
- ***TRN_DB_II:*** This is 2,500,000 word dataset. For more information refer to [1] and [2]. The corpus can be downloaded from [10].
- ***TST_DB:*** This is a 11,000 word test set. For more information refer to [2]. The corpus can be downloaded from [11].
- ***ATB:*** LDC Arabic Tree Bank. For more information refer to [12].

IX. SYSTEM EVALUATION

A. Effect of CSR Method

The objective of this experiment is to show the effect of CSR method described in IV. The test set is TST_DB. Results in Table VI show improvement around 2% absolute in all tested datasets. This represents 17% relative improvement of error.

B. Effect of Class Context Learning

The objective of this experiment is to evaluate the effect of employing sequential class labels model proposed in V. Test set is TST_DB. The results in Table VII show that employing this feature offers 1% to 2% absolute improvement in accuracy over the basic DBN model alone. This represents a 15.6% relative improvement in error.

TABLE VIII
EFFECT OF CONTEXT ON SYNTACTIC DIACRITIZATION

	Accuracy (%)
Context	88.2
No context	84.1

TABLE IX
EFFECT OF LAST CHARACTER FEATURE ON SYNTACTIC DIACRITIZATION

	Accuracy (%)
With last character	88.2
Without last character	84.5

C. Effect of Adding Input Features Context

The context of the word or character is another feature to be employed in the feature vector as described in V. The dataset used for training is TRN_DB_I and for testing is TST_DB. The effect of employing this feature is described in Table VIII. A significant improvement of around 4% absolute is achieved.

D. Effect of Last Character Feature For Syntactic Case

As described in V the identity of the last character of a word is a critical feature for the syntactic diacritization task. The dataset used for training is TRN_DB_I and for testing is TST_DB. Table IX shows the effect of utilizing this feature. A significant error improvement of about 4% absolute is observed with this new feature.

The reason for this strong improvement is that the Arabic language prohibits some diacritics from being placed over some characters. For example fatha on"و" is prohibited phonetically. Also, it favors some diacritics over some character like fatheten on "ا". A rule based system would have set a rule for that, however, in DNN framework, the system is left to learn such rules.

E. Effect of Character Level Encoding of the Word

As described in V the word identity is an important feature for the diacritization task. Some systems heavily rely on this feature like [1] and [2]. There are two possibilities for encoding this feature. The first is an identity representing the index of the word from a vocabulary vector built from a training set, which we refer to as "Word level". However, this could lead to many Out Of Vocabulary (OOV) cases, in addition to a long vector. On the other hand, a word can be encoded as sequence of the identities of its composing characters, which we refer to as "Character level". The dataset used for training is TRN_DB_I and for testing is TST_DB. Table X shows the effect of utilizing this feature. A significant error improvement of about 2% is witnessed with this feature.

"Word level" could lead to many out of vocabulary (OOV) cases, in addition to a long vector. On the other hand, "Character level" is more efficient under the DNN framework to avoid OOV suffered in [1] and [2], because even if a word is not encountered during training, but a similar one with a character less or more was encountered, then a nearly similar activation of the

TABLE X
EFFECT OF CHARACTER LEVEL WORD ENCODING ON SYNTACTIC DIACRITIZATION

Encoding	Accuracy (%)
Character level	88.2
Word level	86.3

TABLE XI
EFFECT OF DATASET SIZE ON SYNTACTIC DIACRITIZATION

Dataset size	Accuracy (%)
100K	83.2
400K	85.4
750K	87.1

TABLE XII
POS TAGGING RESULTS

Group	Accuracy (%)
Group 1	95
Group 2	98
Group 3	99
Overall	97.3

stochastic binary units would be generated, leading to similar result to the most similar word existing in training data set.

F. Effect of Dataset Size

The objective of this experiment is to show the effect of dataset size on the accuracy, see Table XI. The dataset used for training is TRN_DB_I and for testing is TST_DB. Results in show improvement of accuracy with the increase of training set size, however, this trend saturates as the dataset size increases.

G. PoS Tagging Results

In this section the DNN-PoS tagger is tested. The DNN architecture used is 60-60-60. As described in VI, three groups of PoS tags were identified in the TRN_DB_I dataset. Results for each group and overall are shown in Table XII.

H. Comparison to other Systems

The objective of this experiment is to evaluate the performance of the proposed system for Arabic diacritization versus the architecture in [1] and [2], the MaxEnt model proposed in [3] and the MADA system [4]. These systems represent the state of the art in Arabic diacritization systems, with the best reported accuracy in literature. The evaluation was done on all the datasets as explained in [2]. The PoS features are extracted using the DNN-PoS tagger, since the TRN_DB_II / TST_DB dataset contains only raw text without ready PoS features.

Results in Table XIV show that the proposed system achieves improved performance by around 1.2% over the system in [2], which represents 9.23% of the error, evaluated on the (TRN_DB_I + TRN_DB_II / TST_DB) dataset. Also, on the ATB standard dataset, the proposed system achieves 0.9% absolute improvement over the best result in literature.

TABLE XIII
COMPARISON TO HYBRID ARCHITECTURE WITH READY POS FEATURES

System	Syntactical accuracy (%)
Deep network + CSR	90.2
Hybrid Architecture [1]	88.2

Another comparison is done when the dataset TRN_DB_I is used with manual PoS features. Results in Table XIII show that the proposed system achieves better performance by 3.2% over the system in [1], which represents 24.6% of the error. The importance of this experiment is to isolate the automatic PoS tagging errors from the evaluation.

I. Error Analysis

The error analysis presented in this section (see Fig. 10) quantifies the effect of each contribution of features or algorithms in the system. This analysis is conducted for syntactic diacritization.

The baseline system considered is the DNN system described in III with PoS features only. is obtained on a dataset for which PoS are readily available (see VIII TRN_DB_I/TST_DB), which explains the 90.2% reached in the end.

With PoS features only, the accuracy is just 77.9%. Employing the context features of the surrounding words and characters of the current word raises the accuracy by 4.35%. Adding the last character identity to the feature vector adds another 3.95% to the accuracy. The identity of the word feature, encoded as sequence of characters adds another 1%.

Inserting the class labels context enhances the accuracy by 1.1%. Finally, utilizing the CSR algorithm boosts the accuracy by a final 1.9% to reach 90.2%.

From the comparisons in Table XIV and the above analysis of error some conclusions can be drawn about the effect of the underlying classifier used to approach the diacritization problem. While our classifier is a DNN, the one used in [3] is MaxEnt. The baseline accuracy reached with the DNN with PoS features is around 82.25%, which is almost the same as the MaxEnt results.

Adding the last character feature and word identity improves the accuracy to 87% which is almost the same as the result obtained with the hybrid architecture in [1] and [2]. This conclusion is expected because the hybrid architecture depends on word matching compared to a given database and automatically assigns the diacritics based on the pattern matched, in addition to OOV treatment. Adding the word identity and last character feature to our approach results in a similar system to the hybrid architecture, while encoding the word identity as a sequence of characters treats the OOV problem as well, and hence a similar performance is reached.

The CSR method is the final push that boosts our system beyond the best reported result in literature, where automatic confusion analysis is conducted to resolve specific confusions, thus boosting the performance by an extra 1.9%.

X. CONCLUSION

In this paper the problem of Arabic diacritization restoration is tackled under the deep learning framework, taking advantage

TABLE XIV
COMPARISON TO OTHER SYSTEMS

System	Dataset	Syntactical accuracy (%)	Morphological accuracy (%)
Deep network + CSR (This paper)	TRN_DB_I + TRN_DB_II / TST_DB	**88.2**	**97**
	ATB	**88.4**	**97**
Hybrid Architecture – Rashwan et al. [1]	TRN_DB_I + TRN_DB_II / TST_DB	87	96.4
	ATB	87.5	96.2
MaxEnt - Zitouni et al. [3]	ATB	82	94.5
MADA - Habash and Rambow [4]	ATB	85.1	95.2

of DBN model training. As part of the proposed deep system, a PoS tagger for Arabic transcripts using deep networks is proposed as well. The first contribution is the introduction of the Confused Sub-set Resolution (CSR) architecture to enhance the accuracy.

Design of feature vector played a key role in error improvement. Specifically, using features like last character identity had a valuable contribution to error improvement by about 4%. Including class labels context features in auto-regressive fashion also has a good impact of 1.1% on error improvement. Finally, encoding words as sequences of characters reduces OOV cases and enhances the accuracy.

CSR enables resolution of confusions between diacritics. A network-of-networks architecture formed of a group of classifiers, each working to resolve a set of confusions, is directly generated to enhance the overall accuracy by about 2%. The identified confusions and the architecture reflect the grammatical and syntactical rules of Arabic.

Evaluation of the proposed system is made on two different datasets, custom and standard, both of which are available online to enable replicating the experiments. Details of feature vectors formatting and the used features are presented to facilitate replication of results. The standard LDC Arabic Tree Bank dataset is used to bench mark the system against the best three systems in literature, showing that our system outperforms all previously published baselines. The effect of each proposed method is presented separately. Results show improvements ranging from 1.2% to 2.8% over the best reported results representing 22% improvement of the error.

REFERENCES

[1] M. A. A. Rashwan, M. Attia, S. M. Abdou, S. R. Abdou, and A. A. Rafea, "A hybrid system for automatic arabic diacritization," in *Proc. Int. Conf. Nat. Lang. Process. Knowl. Eng.*, Sep. 24–27, 2009, pp. 1–8.

[2] M. A. A. Rashwan, M. A. S. A. A. Al-Badrashiny, M. Attia, S. M. Abdou, and A. A. Rafea, "A stochastic Arabic diacritizer based on a hybrid of factorized and unfactorized textual features," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 1, pp. 166–175, Jan. 2011.

[3] I. Zitouni, J. S. Sorensen, and R. Sarikaya, "Maximum entropy based restoration of arabic diacritics," in *Proc. 21st Int. Conf. Comput. Linguist. and 44th Annu. Meeting Assoc. Comput. Linguist. (ACL); Workshop Comput. Approach. Semitic Lang.*, Sydney-Australia, Jul. 2006.

[4] N. Habash and O. Rambow, "Arabic diacritization through full morphological tagging," in *Proc. 8th Meeting North Amer. Chap. Association Comput. Linguist. (ACL); Human Lang. Technol. Conf.*, 2007, HLT- NAACL.

[5] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, 2006.

[6] R. Salakhutdinov, "Learning deep generative models," Ph.D. dissertation, Graduate Dept. of Comput. Sci., University of Toronto, Toronto, ON, Canada, 2009.

[7] H. Raafat and M. A. A. Rashwan, "A tree structured neural network, ," in *Proc. 2nd Int. Conf. Docum. Anal. Recogn.*, Oct. 20–22, 1993, pp. 939–941.

[8] L. Hai-Son, I. Oparin, A. Allauzen, and J. Gauvain, "Structured output layer neural network language model," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, May 22–27, 2011, pp. 5524–5527.

[9] L. Minh-Thang, S. Richard, and D. M. Christopher, "Better word representations with recursive neural networks for morphology," in *Proc. Conf. Comput. Nat. Lang. Learn. (CoNLL '13)*, 2013.

[10] [Online]. Available: http://www.RDI-eg.com/RDI/Training-Data,TRN_DB_II

[11] [Online]. Available: http://www.RDI-eg.com/RDI/TestData,TST_DB

[12] LDC Arabic Tree Bank Part 3, [Online]. Available: http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2005T20

M. A. A. Rashwan is with The Engineering Company for the Development of Computer Systems (RDI), Egypt, and also with the Department of Electronics and Electrical Communications, Faculty of Engineering, Cairo University, Giza, Egypt.

Ahmad El Sallab has graduated from the Electronics and Communications department at the Faculty of Engineering, Cairo University in 2005 with Excellence Degree with Honor. Since his graduation in 2008, he has been a Teacher Assistant in the department of Mathematics and Physics, Faculty of Engineering, Cairo University. He finished his M.Sc. in 2009 in the area of speech recognition and hardware implementation on FPGA of some parts of the speech recognition systems front-end. He started his Ph.D. program on 2010 in the area of machine learning, where he focused on deep learning applications and algorithms, with special focus on Natural Language Processing on Arabic language. He acquired his Ph.D. degree in 2013. During his academic path, he has published seven papers in four conferences and three journals, and was assigned as designated reviewer in another two conferences. In parallel to his academic career, he has worked in industry for multinational companies like Intel and Valeo as Software Architect and Senior Software Team Leader for more than nine years.

Hazem M. Raafat received the B.Sc. (Hons.) degree in electronics and communications engineering from Cairo University, Egypt, in 1976, and a Ph.D. degree in systems design engineering from the University of Waterloo, Canada, in 1985. He was an Associate Professor with the Department of Computer Science at the University of Regina, Canada where he also held a joint appointment with the Electronics Information Systems Engineering Department. He is currently with the Computer Science Department at Kuwait University. His research interests include data mining, computer vision, pattern recognition, multiple-classifier systems, texture analysis, and natural language processing. He is a member of IEEE and ACM.

Ahmed Rafea obtained his Ph.D. from University Paul Sabatier, in Toulouse, France, and is currently a Professor and Chair of the Computer Science and Engineering Department at the American University in Cairo. He was the Principal Investigator of many projects on machine translation, text mining, sentiment analysis, and knowledge engineering in collaboration with American and National Universities and Institutions. He has authored over 180 conference proceedings, book chapters, and scientific papers in both international and national journals. His research interests include natural language processing, machine translation, knowledge engineering, knowledge discovery, and data, text and web mining.