Deep Learning Models for Sentiment Analysis in Arabic

Ahmad A. Al Sallab
Electrical and Communication Department,
Faculty of Engineering, Cairo University, Egypt
ahmad.elsallab@gmail.com

Ramy Baly, Gilbert Badaro, Hazem Hajj
Electrical and Computer Engineering Department,
American University of Beirut, Lebanon
{rgb15, ggb05 hh63}@aub.edu.lb

Wassim El Hajj
Computer Science Department,
American University of Beirut, Lebanon
we07@aub.edu.lb

Khaled B. Shaban
Computer Science and Engineering Department,College of Engineering,
Qatar University
khaled.shaban@qu.edu.qa

Abstract

In this paper, deep learning framework is proposed for text sentiment classification in Arabic. Four different architectures are explored. Three are based on Deep Belief Networks and Deep Auto Encoders, where the input data model is based on the ordinary Bag-of-Words, with features based on the recently developed Arabic Sentiment Lexicon in combination with other standard lexicon features. The fourth model, based on the Recursive Auto Encoder, is proposed to tackle the lack of context handling in the first three models. The evaluation is carried out using Linguistic Data Consortium Arabic Tree Bank dataset, with benchmarking against the state of the art systems in sentiment classification with reported results on the same dataset. The results show high improvement of the fourth model over the state of the art, with the advantage of using no lexicon resources that are scarce and costly in terms of their development.

1 Introduction

With the revolution of web 2.0 and the amount of opinionated data generated by online users, personal views and opinions are no longer constrained to authors in newspapers or custom opinion surveys. Instead, almost anyone can express opinions through social media. The abundance of these opinions and their availability and accessibility gave birth to automated applications that use sentiment analysis (opinion mining) as a key factor in predicting stock market, evaluating products, surveying the public, etc. However, automated sentiment analysis is still far from producing output with quality comparable to humans due to the complexity of the semantics. Furthermore, the Arabic language adds another dimension of difficulty to automated sentiment analysis due to its morphological richness, ambiguity, and the large number of dialectal variants. These challenges add to the complexity of the required natural language processing (NLP).

Many methods have been suggested in literature to address automated sentiment analysis. One of the prominent approaches is the use of machine learning (ML) techniques, where sentiment analysis is formalized as a classification task. The predicted classes are typically chosen to be positive or negative sentiment. The classification tasks range from classifying the sentiment of words, phrases, sentences, or sometimes documents. Deep learning has been recently considered for sentiment analysis (Socher et al. 2013). Socher et al. 2013 worked on phrase level sentiment classification using the Recursive Neural Tensor Network (RNTN) over a fine grained phrase level annotated corpus (Stanford Sentiment Tree Bank). Other deep learning models that can potentially be used in sentiment analysis include deep neural networks (DNN), convolutional neural networks (CNN) (LeCun et al. 1995), Deep Belief Networks (DBN) with fast inferencing of the model parameters (Hinton et al. 2006), and recurrent neutral network (RNN) (Socher et al. 2013).

We aim in this work to investigate the merit of using deep models for sentiment analysis in Arabic, focusing on the sentence level sentiment classification. To the best of our knowledge, this is the first attempt to explore deep learning models for sentiment classification in Arabic. For the vector space representation of text, we utilize ArSenL (Badaro et al. 2014), a recently published sentiment lexicon. Each word in the lexicon is associated with three sentiment scores indicating levels of positivity, negativity, and neutrality. ArSenL includes 28,780 Arabic lemmas with the corresponding number of 157,969 synsets. We explore four deep learning models: DNN, DBN, Deep Auto Encoder (DAE), and combined DAE with DBN. DNN applies back propagation to a conventional neural network, but with several layers. DBN applies generative pre-training phase before feeding a discriminative fine tuning step. DAE provides a generative model representation for the original but with reduced dimensionality. Finally, the RAE aims at parsing the raw sentence words in the best order that minimizes the reconstruction error of re-generating the same sentence words in the same order; in other words, it aims at discovering the best parse tree that maximizes the probability of the input data.

Both DAE and RAE models aim at providing a compact representation of the input sentence. Both models are based on unsupervised learning, where their objective is the minimization of reconstruction error of the input, so no manual annotation is needed. The main difference is that; RAE considers the context and order of parsing of the sentence. This recursion enables parsing variable length sentences. While the DAE is parsing the whole sentence words at once in the first layer, with no consideration of the order of parsing of words, and keep feeding the representation forward in the deep architecture on the hope that useful features are extracted at each layer of depth. This property makes it mandatory to have fixed length features vector, which promotes the Bag-of-Words (BoW) model.

Both DAE and RAE models require a classifier on top of their obtained representation. In case of DAE, the classifier is the DBN, while in case of RAE, the classifier is a softmax layer.

The Linguistic Data Consortium Arabic Tree Bank (LDC ATB) dataset is used to evaluate the proposed models. The input data to the first three models depend on the BoW model, with the utilization of lexicon scores. In our case it is ArSenL, as special sentiment features.

The rest of the paper is organized as follows: Section 2 overviews the work related to sentiment classification in Arabic. Section 3 describes the features employed from ArSenL. Section 4 includes a description of the proposed deep learning models. Section 5 presents the results of the evaluation on LDC ATB, and section 6 concludes the paper.

2 Related Work

This section presents an overview of different approaches proposed to perform opinion mining in Arabic focusing on practices pertaining to preprocessing, feature engineering, modeling, and evaluation methods.

Word n-grams are considered the most common features that have been used, with different preprocessing and representation settings, to train classification models. In general, using higher-order n-grams (bigrams and trigrams) – represented with term-frequency inverse-document-frequency (TFiDF) weights achieved better results compared to unigrams (Rushdi et al. 2011, Mountassir et al. 2012). These features were used to train different classification models with support vector machines (SVM) achieving better performances (Rushdi et al. 2011, Aly and Attiya 2013, Al-Kabi et al. 2013, Shoukry et al. 2013) with a few exceptions where Naïve Bayes was found superior (Mountassir et al. 2012, Elawady et al. 2014). Ensemble techniques were also utilized for additional performance improvement (Omar et al. 2013). The impact of stylistic features was introduced in (Abbasi et al. 2008). These features were found beneficial when used along with syntactic features.

Arabic sentiment lexicons are also used to engineer features. Examples are ArSenL (Badaro et al. 2014), SIFAAT (Abdul-Mageed et al. 2011) and ArSeLEX (Ibrahim et al. 2015). Deep learning models have recently gained popularity, and can potentially be used in sentiment analysis. These models include DNN, CNN (LeCun et al. 1995), DBN, DBN with fast inference of the model parameters (Hinton et al. 2006), and RNN (Socher et al. 2013). Recently, Socher et al. (Socher et al. 2013) worked on phrase level sentiment classification in English using Recursive Neural Tensor Networks over a fine grained phrase level annotated corpus (Stanford Sentiment Tree Bank).

Raw sentence	فالمهم هو التوصل الى اتفاق جيد					
	جيد	اتفاق	الى	التوصل	هو	فالمهم
Binarized input (variable length)	1398	1045	24	256	43	103
Semantic word embedding representation	$Lb_{\text{جيد}} \in \Re^N$	$Lb_{\text{اتفاق}} \in \Re^N$	$Lb_{\text{الى}} \in \Re^N$	$Lb_{\text{التوصل}} \in \Re^N$	$Lb_{\text{هو}} \in \Re^N$	$Lb_{\text{فالمهم}} \in \Re^N$
RAE representation	$\hat{x} \in \Re^N$					

Table 1 Example of parsing a sentence from its raw words into their embedding representation

Lastly, a variety of corpora have been used for evaluation such as OCA (Opinion Corpus for Arabic) (Rushdi-Saleh 2011), LABR (Large-scale Arabic Book Reviews) (Mountassir et al. 2012), sentences from the Penn Arabic Treebank (PATB) part 1, version 3.0 (Abdul-Mageed et al. 2011) and many other self-created corpora.

3 Data Feeding Deep Learning

In the first three deep learning models (DNN, DBN and DAE), we employ features based on ArSenL where the words in each sentence are represented in a vector of length equal to the number of entries in the lexicon. Instead, of using TFIDF scores or binary representations of the words, we focus the evaluation on the impact of sentiment lexicon features due to their demonstrated relevance in past literature. In ArSenL, there are 3 scores for each lemma (denoting positive, negative and neutral polarity). The sum of the 3 scores adds up to 1. As a result, the feature vector will be three times the size of the selected text in the corpus. For the LDC ATB dataset, 3795 entries are matched to in ArSenL, resulting in a feature vector of length 11385. It is worth noting that this vector representation is sparse, and we refer to it as *arsenl_lemma*. We also use aggregated sentiment score for the whole sentence, thus obtaining three scores per sentence for positive, negative and neutral polarities. In this case, the feature vector is of length three and we refer to it as *arsenl_sentence*.

For the forth and last model (RAE), the input is the raw words indices that constitute each sentence, hence, the length of input is variable per sentence. The words' indices are drawn from a known vocabulary obtained from a separate and independent training set. Test set words that are not encountered in training are considered "UNKNOWN" and are given a special index. Stop words are not removed.

For the RAE, the main preprocessing steps are:

1) Vocabulary vector build: parse the whole dataset to obtain the encountered vocabulary words. No stop words removal or stemming is done.
2) Each sentence is represented as list of word indices $b_{word} \in \Re^{|V|}$, where $|V|$ is the size of the vocabulary, in our case for the LDC ATB dataset, it is 31850 words. Each word in a sentence is looked up in the table $L \in \Re^{N x|V|}$ where N is the size of the resulting embedding representation vector (in our experiments it is set to 50)
3) The resulting sequence of representations is fed forward in the parse tree of the RAE to obtain one representation $\hat{x} \in \Re^N$ for the whole sentence.

An example of a parsed sentence is described in Table 1

4 Deep Learning Models for Sentiment Analysis in Arabic

Three models are proposed under deep learning framework: DNN, DBN, and a combined Auto Encoder with DBN. The network architecture in terms of depth, breadth, and hyper parameters settings are set based on the recommendations in (Bengio et al. 2009) and (Bengio et al. 2012).

For the DNN architecture, the number of neurons in each layer is selected to yield the best accuracy for a selected development data set. For the considered data, the number of neurons came out at, 40 per layer. The depth of the DNN network is selected by iteratively incrementing the number of layers one at a time while evaluating the accuracy at every increment. The depth of 3 layers was found to yield the best accuracy with the selected data set. A decision softmax layer composed of two neurons was then added on top of the three network layers. For training the model, we used supervised back propagation. The objective of the model is to minimize the error of the network output versus the true sentiment class label for each training case. The remaining settings for DNN model are: (1) conjugate gradient algorithm is used for gradient updates with three line searches; (2) weights are randomly initialized from Gaussian distribution of 0 mean and standard deviation of 1; and (3) the activation function of each neuron is taken as hyperbolic tangent activation. Training is conducted in batches of size 100 cases for 50 epochs. The resulting architecture of the DNN model is shown in Figure 1.

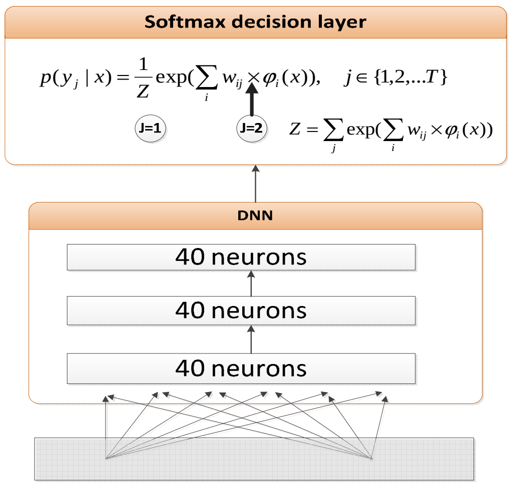

Softmax decision layer
$p(y_j \mid x) = \frac{1}{Z}\exp(\sum_i w_{ij} \times \varphi_i(x)), \quad j \in \{1,2,...T\}$
J=1
J=2
$Z = \sum_j \exp(\sum_i w_{ij} \times \varphi_i(x))$
DNN
40 neurons
40 neurons
40 neurons

Figure 1. DNN Architecture

The second model is based on the DBN model described in (Hinton et al. 2006). The learning process is performed in two phases. First, a generative unsupervised pre-training phase is developed based on stacked Restricted Boltzmann Machine wake-sleep algorithm at each layer (Hinton et al. 2006). In the second phase, the weights of the network are used to initialize a discriminative supervised model similar to DNN. The difference with the conventional DNN is the addition of the pre-training phase, which was found to avoid model over fitting (Bengio et al. 2012). The same network architecture of DNN is used for both, pre-training and fine-tuning phases. Both phases undergo 50 epochs of weights updates. The resulting DBN model architecture is depicted in Figure 2.

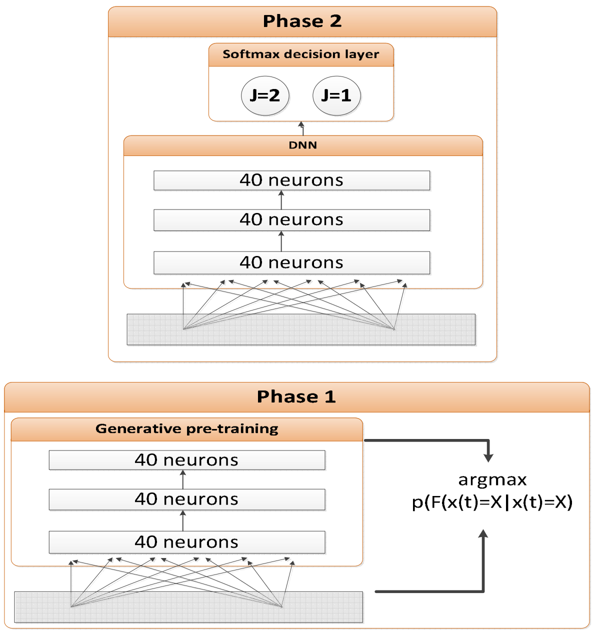


Figure 2. DBN Architecture

For the third model, a generative deep auto encoder model is first trained with the objective of minimizing the error between the applied and the reconstructed vectors. The result of the auto-encoder is then followed by a model similar to the DBN model, with pre-training and fine tuning phases. The error function is taken as the difference between the applied features vector and the reconstructed vector in the reverse order of the deep auto encoder. The auto encoder architecture is taken as 100-50-20 in three respective layers. The idea is to obtain a reduced dense dimension vector with accurate representation of the input data. Since the input vector is sparse, we cannot directly consider its dimension as the real dimension representing the input data, as it contains many zeros. Hence, we consider the 40 neurons, which were taken in the first two models as the hidden layers dimensions, and we target 50% reduction in the deep auto encoder. To achieve this reduction ratio, we start at 100 neurons and reduce the number of neurons by 50% as we go deeper in the model. The resulting architecture is shown in Figure 3. The figure shows the unfolded architecture of the employed encoder. The encoded data representing the input is taken from the third activation layer. The reconstructed output is then taken from the 6th layer, which is equivalent to the 1st layer by symmetry of the proposed architecture.

After the deep auto encoder is derived, the training data is fed to the encoder to obtain the representative 20 dimension codes for each entry of the dataset. The new obtained codes are then used as training data for another DBN. This time, no pre-training is run in the DBN model since pre-training already happened during the deep auto encoder training. The obtained 20 dimension vectors are dense, unlike the original feature vectors of the training set. Hence different architecture needs to be employed in the DBN to account for different combinations of the dense data inside the code vectors. The best architecture of the layers for the DBN in this case was found to be three layers with 400 neurons in each layer

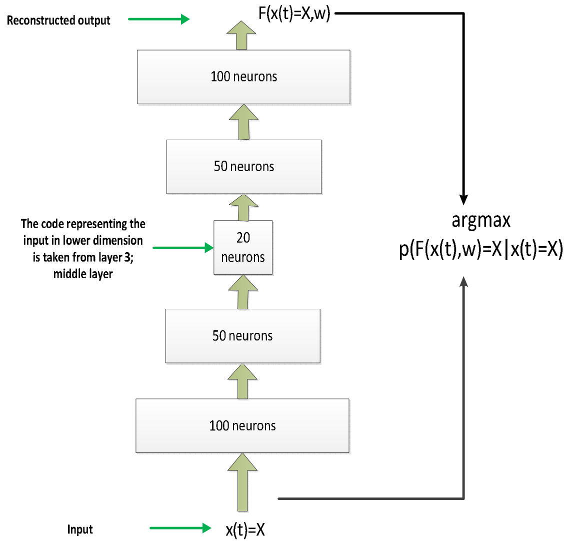

Reconstructed output
F(x(t)=X,w)
100 neurons
50 neurons
The code representing the input in lower dimension is taken from layer 3; middle layer
20 neurons
argmax p(F(x(t),w)=X|x(t)=X)
50 neurons
100 neurons
Input
x(t)=X

Figure 3 DAE Architecture.

So far, the input data to the first three models depend on the BoW, with the utilization of lexicon scores, in our case it is ArSenL, as special sentiment features. The BoW suffers two main issues: 1) Poor representation of features, which generates sparse vectors of words, where most of its encoded information and features are not relevant to the classification of the current case at hand. This sparseness hurts the reconstruction of the DAE and causes high errors resulting in poor representations of the input. 2) No consideration of context, where the words are encoded irrespective to their order in the original sentence. The BoW model renders the lexicon scores useless, and sometimes misleading, because it draws them out of context. In other words, a word cannot be absolutely positive or absolutely negative. However, the sentiment of a word is usually context dependent. For example, the word “beat” is usually a negative word. However, in the context of “We have beaten the other team”, it becomes a positive one. Also, positivity and negativity of a word is perspective dependent. For example: “team A has beaten team B” is a positive context for team A, but negative for team B. This renders the absolute lexicon scores useless or even misleading in some cases if taken without the consideration of the context.

The fourth and last model is the Recursive Auto Encoder (RAE). The RAE is a member of the recursive family of deep learning models (Socher 2011, 2013). The main advantage of the RAE is that; it is unsupervised, so it does not require parse tree annotation like other members of this family, like Recurrent Neural Networks (RNN). The basic block of the RAE is the normal auto encoder described earlier, where the objective is to minimize the error between the original raw input vector and the reconstructed one, called the reconstruction error. They have been used for dimensionality reduction and hash space search.

In NLP, the input is usually the BoW vector, with 1’s at the positions where a word of the vocabulary is encountered in the current sentence at hand, leaving many irrelevant zeros at the rest of the vector positions. This hurts badly the reconstruction capability of the AE and makes its convergence harder. Also, no context is captured in this model. To address this, the second component of the RAE model is the recursion parse tree, where the sentence words are parsed/visited in a certain order that captures their semantic meaning, and how they influence and sometimes inflect the meanings of each other. For example, the meaning of “good” is the opposite of “not good”.

The basic block of recursion is the AE, which is a binary encoder in our case. The goal of auto encoders is to learn a representation of their inputs. The algorithm in Socher et al. 2013 is described in brief here. At each step of parsing, the weights of the basic AE are updated so as to minimize the reconstruction error (see Figure 5). However, this procedure assumes that the parse tree order is known, which is not. A prior step is required to discover the best parse tree first. This is done through a greedy breadth first algorithm. At each recursion step, all the possible remaining words of the sentence are attempted; generating a representation and associated reconstruction error. The next node to be included in the tree is the one that generated the minimum reconstruction error. This algorithm is greedy because it considers the best solution at the current step without considering the global situation, which simplifies calculations and reduces the processing time.

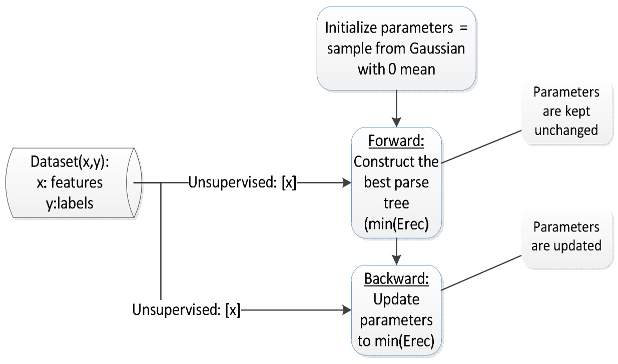

Initialize parameters = sample from Gaussian with 0 mean
Parameters are kept unchanged
Dataset(x,y): x: features y:labels
Unsupervised: [x]
Forward: Construct the best parse tree (min(Erec)
Parameters are updated
Unsupervised: [x]
Backward: Update parameters to min(Erec)

Figure 4 Training procedure of RAE with greedy discovery of the best parse tree

So the RAE learning process with tree structure discovery shall go as follows (shown in Figure 4):

1) Initialize the RAE weights and word embeddings L with zero mean Gaussian samples
2) Forward path:
    a. Initialize the parents node list to null
    b. At each step, try all possible extension leaves to the tree from the list of all candidate leaves
    c. For each extension, evaluate the reconstruction error
    d. Choose the leaf that minimizes the error and add it to the parents list
3) Backward path:
    a. Once the tree is constructed, the weights of RAE can be adjusted same as done in the normal training of the AE described
4) Repeat 2 and 3 for each training case

At this step, we have obtained a RAE that is able to provide a sentence wide representation by recursively parsing its words in the best order. However, to build a sentiment classifier using this learnt network two components are missing. The first one is concerned with handling the raw input words, and obtaining a good representation out of it that encodes the context of the word. This is often referred to as the word embedding. In our approach, this block is implemented as a lookup table $L \in \Re^{Nx|V|}$ where N is the size of the resulting embedding representation. In our setting, this block is initialized by sampling it from a zero mean Gaussian distribution. During the learning process described, the weights of this block are learnt from the unsupervised data.

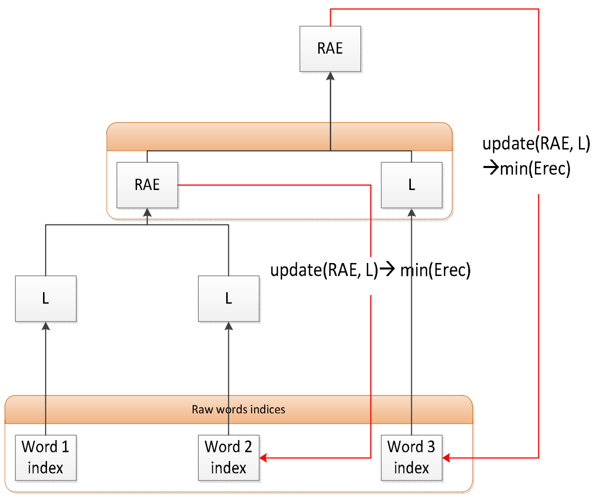


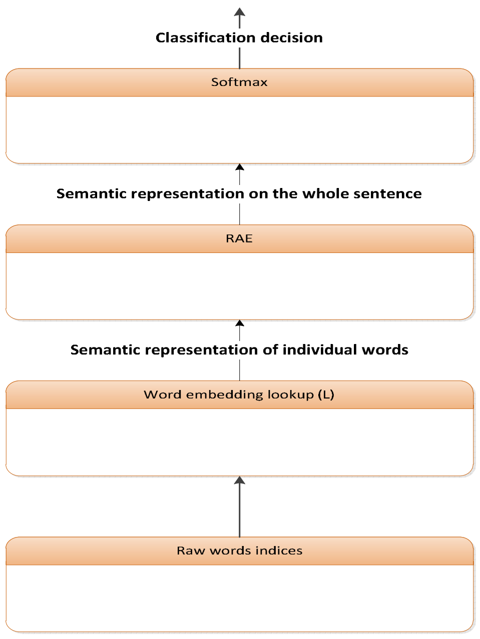
Figure 5 Word embedding matrix update in the training process of RAE.

The other missing block, is the classifier on top of the RAE representation. This is the supervised part of the system, where it is trained based on the sentiment annotations given to each training case. It could be any supervised machine learning classifier. In our case, it was a simple softmax layer.

The full architecture of the system is shown in Figure 6. The main steps are as follows:

1) Build the word embedding matrix L.
    a. The input are the raw sentences represented as sequence of its constituting words indices.
    b. The output is the semantic representation, i.e. the result of look up of each word index in the matrix L.
2) Construct the RAE parse tree and update its weights for best reconstruction.
    a. The input is the sequence of semantic representations obtained from the word embedding block.
    b. The output is the top level compact representation of the parsed sentence.
3) Train a classifier on top of the RAE representations. In our case this is just a softmax (MaxEnt) layer.
    a. The input is the representation obtained from the RAE
    b. The output is the classification decision. In our case; positive or negative.

The RAE model has the following advantages: 1) The phase of RAE construction and parse tree discovery is completely unsupervised, while the only supervised part is the fine tuning phase. This property enables the adaptation and enhancement of the system on any un-annotated dataset. 2) The input is completely raw words indices, with no lexicon required, which are a hard to build language resource in terms of effort and cost.



Classification decision
Softmax
Semantic representation on the whole sentence
RAE
Semantic representation of individual words
Word embedding lookup (L)
Raw words indices

Figure 6 Sentiment classification using RAE

The problem of context handling is partially solved by the RAE model, where the order of parsing is variable with each new sentence, and hence a different representation is obtained for each sentence according to its semantics. However, as far as the task of sentiment classification is concerned, the sentimental context is not captured, but the semantic context is depicted. In other words, the parse tree is discovered according to which n-grams sequence are valid or form a meaningful constituent, and hence the parse tree is formed. However, the sentiment context of such n-grams is not considered. To tackle this issue, a different parse tree is needed; a sentiment parse tree. An example of which, is the Stanford sentiment Tree Bank (Socher et al. 2013), which requires a huge and specialized annotation effort for the whole parse tree of the sentence not the overall sentence sentiment. The classification is then based on RNN. This is considered as a future work due to the unavailability of such sentiment tree bank as a required language resource for this type of networks.

5 Evaluation

To evaluate the models, LDC ATB dataset is used for training and testing. The dataset is split into 944 training sentences and 236 test sentences. Only positive and negative classes are considered for the data represented by *Arsenl_lemma*, and *Arsenl_sentence* features separately.

The results in Table 2 show that both, DBN and Auto Encoder (models 2 and 3) do not suffer over fitting while model 1 does. This is in line with the observation in (Bengio et al. 2012) which indicates that pre-training provides kind of regularization on the learned weights of the network. This is expected because deep auto encoder output provides good generalization of the input data, and has even less tendency to over-fit training data. With selected architectures, and in most cases, the F1 measures were close to SVM, and sometimes superior. The accuracy measures were not superior.

The input representation in the first three models is based on the BoW encoding of the ArSenL scores, which makes the features vector very sparse with too many zeros. This hurts badly the reconstruction capability of the network, because slight errors around zeros add up. This effect is reduced when the features vectors are first fed to a DAE to obtain a compact representation rather than a sparse one.

A better representation would be to select only the vocabulary words that are encountered in the sentence under focus. However, this will make the features vector length variable. A recursive model addresses this problem by parsing the sentence words recursively to obtain sentence wide representation considering only the vocabulary words that exist in the sentence. This is one of the reasons why the RAE is superior to the other three models.

The RAE model outperforms all the other models by a large margin of around 9%. As pointed out earlier, in this model, semantic context and the parsing order of words are considered. In the same time, no lexicon is used, and no special features are used, but only raw words as input. Table 3 shows the result of benchmarking the deep learning models proposed against other systems in literature, like linear SVM applied to ArSenL scores (Badaro el al. 2013) and SIFAAT (Abdul-Mageed et al. 2011), which represent the state of the art results on the LDC ATB dataset in Arabic sentiment classification.
RAE outperformed SIFAAT by around 14%, while it outperformed linear SVM on ArSenL scores by around 9%.

	RAE		*Linear SVM*		*DNN (model 1)*		*DBN (model 2)*		*Deep auto – DBN (model 3)*	
Feature	*Accuracy (%)*	*F1 score (%)*	*Accuracy (%)*	*F1 score (%)*	*Accuracy (%)*	*F1 score (%)*	*Accuracy (%)*	*F1 score (%)*	*Accuracy (%)*	*F1 score (%)*
Arsenl_lemma	-	-	66.1	59.2	55.5	44.5	57.5	46.8	60.4	60.5
Ar-senl_sentence	-	-	61.4	62.8	53.4	44.3	53.4	40.1	56.1	43.5
Raw words	**74.3**	**73.5**	45.2	44.1	39.5	39.1	41.3	40.5	43.5	43.7

Table 2. Evaluation results on LDC ATB

	RAE	*SIFAAT*	*Linear SVM - ArsenL*	*DNN (model 1)*	*DBN (model 2)*	*DAE – DBN (model 3)*
Average F1 score (%) (Pos/Neg)	**73.5**	59.2	64.5	44.5	46.8	60.5

Table 3. Benchmark results on LDC ATB

In our experiments on RAE we focus on the idea of obtaining a representation that takes into consideration the context of the word. At the same time, we want to take advantage of the unsupervised nature of RAE that avoids the use of sentiment lexicon. Another future direction will be to consider using ArSenL lexicon to create better representation of word embeddings. This can be done by creating special word embedding blocks with the objective of generating the ArSenL sentiment scores, and then use this representation as input to the RAE. This is considered as a pre-training step to the embedding block rather than random initialization or n-gram validity task. Also, the pre-training using ArSenL enables the consideration of the individual words sentiment in addition to the semantic words context.

6 Conclusion

In this paper, a deep learning approach is proposed for the sentiment classification problem on Arabic text. Three architectures were proposed and derived for: DNN, DBN and Deep Auto Encoders. The features vector used the sentiment scores from ArSenL lexicon. LDC ATB dataset was used to evaluate the models, comparing their accuracy and F1 scores. It was found that, Deep Auto encoder model gives better representation of the input sparse vector. We also proposed a forth model, RAE, which was the best deep learning model according to our results, although it requires no sentiment lexicon. The results show around 9% improvement in average F1 score over the best reported results in literature on the same LDC ATB dataset in the sentiment classification task for Arabic.

Future work includes: 1) the enhancement of the word embedding block by employing large unsupervised corpus, and 2) enhancing the way the parse tree is constructed by improving the search method and its objective, so that it could be more directed towards semantic and syntactic correctness of the resulting parse tree, rather than depending on the reconstruction error alone.

Acknowledgement

This work was made possible by NPRP 6-716-1-138 grant from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

Gilbert Badaro, Ramy Baly, Hazem Hajj, Nizar Habash, and Wassim El-Hajj. "A Large Scale Arabic Sentiment Lexicon for Arabic Opinion Mining." ANLP 2014 (2014): 165.

Socher, Richard, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. "Recursive deep models for semantic compositionality over a sentiment treebank." In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1631-1642. 2013.

Socher, Richard, et al. "Semi-supervised recursive autoencoders for predicting sentiment distributions." Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2011.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In Proceedings of ICML, pages 160–167.

Abdul-Mageed, M., Diab, M. and Korayem, M. (2011). Subjectivity and sentiment analysis of modern standard Arabic. *In Proceedings of the 49th Annual Meeting of the Association for Com-putational Linguistics: Human Language Technol-ogies: short papers*-Volume 2. Association for Computational Linguistics.

Abdul-Mageed, M., & Diab, M. (2012). Toward building a large-scale Arabic sentiment lexicon. In *Proceedings of the 6th International Global WordNet Conference* (pp. 18-22).

G. E. Hinton; S. Osindero; Y. Teh, "A fast learning algorithm for deep belief nets" Neural Computation, vol. 18, pp. 1527–1554, 2006.

Ruslan Salakhutdinov. 2009."Learning Deep Generative Models" PhD thesis, Graduate Department of Computer Science, University of Toronto.

Larochelle, Hugo, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. "Exploring strategies for training deep neural networks." The Journal of Machine Learning Research 10 (2009): 1-40.

Bengio, Yoshua. "Practical recommendations for gradient-based training of deep architectures." In Neural Networks: Tricks of the Trade, pp. 437-478. Springer Berlin Heidelberg, 2012.

LeCun, Yann, and Yoshua Bengio. "Convolutional networks for images, speech, and time series." The handbook of brain theory and neural networks3361 (1995): 310.

Rushdi-Saleh, M., Martín-Valdivia, M. T., Ureña-López, L. A., & Perea-Ortega, J. M. "OCA: Opinion Corpus for Arabic." Journal of the American Society for Information Science and Technology 62.10, 2045-2054, 2011

Rushdi-Saleh, M., Martín-Valdivia, M. T., Ureña-López, L. A., & Perea-Ortega, J. M. "Bilingual Experiments with an Arabic-English Corpus for Opinion Mining." Proceedings of Recent Advances in Natural Language Processing, 2011

Mountassir, Asmaa, Houda Benbrahim, and Ilham Berrada. "A Cross-study of Sentiment Classification on Arabic Corpora." Research and Development in Intelligent Systems XXIX, Springer London, 2012

Aly, M. A., & Atiya, A. F. "LABR: A Large Scale Arabic Book Reviews Dataset." Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, 2013

Elawady, Rasheed M., Sherif Barakat, and Nora M. Elrashidy "Sentiment Analyzer for Arabic Comments", International Journal of Information Science and Intelligent System, 3(4): 73-86, 2014

Al-Kabi, Mohammed N., Nawaf A. Abdulla, and Mahmoud Al-Ayyoub "An Analytical Study of Arabic Sentiments: Maktoob Case Study." Proceedings of the 8th International Conference for Internet Technology and Secured Transactions (ICITST), IEEE, 2013

Shoukry, A., & Rafea, A. "Sentence-level Arabic Sentiment Analysis." Proceedings of International Conference on Collaboration Technologies and Systems (CTS), IEEE, 2012

Omar, N., Albared, M., Al-Shabi, A., & Al-Moslmi, T. "Ensemble of Classification Algorithms for Subjectivity and Sentiment Analysis of Arabic Customers' Reviews." International Journal of Advancements in Computing Technology, 14(5), 2013

Abbasi, Ahmed, Hsinchun Chen, and Arab Salem. "Sentiment Analysis in Multiple Languages: Feature Selection for Opinion Classification in Web forums." ACM Transactions on Information Systems (TOIS) 26.3, 2008

Ibrahim, Hossam S., Sherif M. Abdou, and Mervat Gheith. "Sentiment Analysis for Modern Standard Arabic and Colloquial." International Journal on Natural Language Computing (IJNLC) Vol. 4, No.2, April 2015

Abdul-Mageed, M., & Diab, M. T. " Subjectivity and Sentiment Analysis of Modern Standard Arabic." Proceedings of the 5th Linguistic Annotation Workshop, Association for Computational Linguistics, 2011

Nabil, Mahmoud, Mohamed Aly, and Amir Atiya. "LABR: A Large Scale Arabic Sentiment Analysis Benchmark", 2015

ElSahar, H., and El-Beltagy S.R. "Building Large Arabic Multi-domain Resources for Sentiment Analysis." Computational Linguistics and Intelligent Text Processing. Springer International Publishing, 2015

Arabic Tree Bank Part 3, http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=2005T20