AraDIC: Arabic Document Classification using Image-Based Character Embeddings and Class-Balanced Loss

Mahmoud Daif, Shunsuke Kitada, Hitoshi Iyatomi
Hosei University
Graduate School of Science and Engineering
Department of Applied Informatics
{mahmoud.daif.8h@stu., shunsuke.kitada.8y@stu., iyatomi@}
hosei.ac.jp

Abstract

Classical and some deep learning techniques for Arabic text classification often depend on complex morphological analysis, word segmentation, and hand-crafted feature engineering. These could be eliminated by using character-level features. We propose a novel end-to-end Arabic document classification framework, Arabic document image-based classifier (AraDIC), inspired by the work on image-based character embeddings. AraDIC consists of an image-based character encoder and a classifier. They are trained in an end-to-end fashion using the class balanced loss to deal with the long-tailed data distribution problem. To evaluate the effectiveness of AraDIC, we created and published two datasets, the Arabic Wikipedia title (AWT) dataset and the Arabic poetry (AraP) dataset. To the best of our knowledge, this is the first image-based character embedding framework addressing the problem of Arabic text classification. We also present the first deep learning-based text classifier widely evaluated on modern standard Arabic, colloquial Arabic and classical Arabic. AraDIC shows performance improvement over classical and deep learning baselines by 12.29% and 23.05% for the micro and macro F-score, respectively.

1 Introduction

Arabic is one of the six official languages of the United Nations and the official language of 26 states. It is spoken by as many as 420 million people making it the fifth most popular language worldwide. According to the Internet World Statistics, as of 2017, Arab users represent 4.8% of internet users[1].

Arabic can be classified into three different types each having its own purpose and morphology. The modern standard Arabic, the colloquial or dialectal Arabic and the classical or old Arabic. The modern standard Arabic is the official language used in media, government, news papers and is taught in schools. Colloquial Arabic varies between countries and regions. Old or classical Arabic survives nowadays in religious scriptures and old poetry.

Arabic has 28 basic letters all are consonants except three, which are long vowels. Arabic is written from right to left. Most Arabic letters have more than one written form depending on their position in the word. For example, “ س ”. “ ـس ”, “ ـسـ ”, and “ سـ ” are all different forms of the letter “ س ”(sīn). In addition, diacritical marks/short vowels that contribute to the phonology of Arabic, greatly alter the character shape. Example, “ بً ”, “ بٌ ”, “ بٍ ”, “ بّ ”, “ بْ ”, “ بُ ”, “ بَ ”, and “ بِ ” are combination of the letter “ ب ”(bā’) with different diacritics. This visual nature of the Arabic letters is the main motivation for us to use image based embeddings.

The importance of text classification has increased due to the increase of textual data on the internet as a result of social networks and news sites. Common examples of text classification are sentiment analysis (Ibrahim et al., 2015), spam detection (El-Halees, 2009) and news categorization (Shehab et al., 2016). Arabic text classification is particularly challenging because of its complex morphological analysis.

Most research on Arabic text classification has used classical techniques for feature extraction (Salloum et al., 2018), which require complex morphological analysis, such as negation handling(Al-Twairesh et al., 2016), part of speech tagging (Khoja, 2001), stemming (Al-Kabi et al., 2015), and segmentation (Abdelali et al., 2016). Arabic segmentation is especially complex because Arabic words are not always separated by white spaces. It also includes some hand-crafted features

[1] Arabic Speaking Internet Users and Population Statistics. https://www.internet-worldstats.com/stats19.html Accessed: 16-Dec-2018,

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, pages 214–221
July 5 - July 10, 2020. ©2020 Association for Computational Linguistics

like document term matrix with term frequency inverse document frequency (TF-IDF) scores or word count.

Arabic text classification have been often done using classical algorithms like support vector machines (SVMs) or Naive Bayes (Salloum et al., 2018). Despite advances of text classification using deep learning techniques, little work has been done on Arabic. Soliman et al. (2017) introduced AraVec, which is a pretrained distributed word embeddings (Mikolov et al., 2013). They trained their model using the skip-gram and continuous bag of words techniques. They used data from different sources like Wikipedia and Twitter. More recently, Sagheer and Sukkar (2018) used AraVec's pretrained word embeddings with sentence convolutional neural network (CNN) originally proposed by Kim (2014) for Arabic document classification. This method still did not mitigate the problem of Arabic word segmentation.

Those combinations left two major issues unaddressed. First, performance highly depends on morphological analysis and word segmentation, which is difficult for Arabic. The same problem has been addressed for languages such as Japanese and Chinese (Peng et al., 2003). Second, obtaining appropriate embedding (i.e. building hand-crafted features) is difficult.

To solve these problems, character-based approaches utilizing deep learning methods mainly used in image processing have been proposed (Zhang et al., 2015; Shimada et al., 2016; Kitada et al., 2018).

Zhang et al. (2015) introduced a character-level CNN (CLCNN) that treats text as a raw signal at character level. The CNN then learns the language morphology and extracts appropriate features for text classification. Their method mitigated the issue of complex morphological analysis.

After that, Shimada et al. (2016) proposed image-based character embeddings for Japanese and Chinese text classification. Their model was composed of a convolutional auto-encoder (CAE) (Masci et al., 2011) and a CLCNN. They were the first to handle a character as an image and obtained character-embedding with their CAE. They also introduced wild card training as a data augmentation technique, which is dropout (Srivastava et al., 2014) on the embedding space.

Later, Liu et al. (2017) used image-based character embeddings learned through a character encoder (CE) to train a gated recurrent unit (GRU) for Japanese, Chinese, and Korean text classification.

Kitada et al. (2018) proposed CE-CLCNN that concatenated Liu et al. (2017)'s CE with CLCNN as an end-to-end system and introduced random erasing on image domain as a data augmentation method. These models using character-level features learn language morphology eliminating the need for complex morphological analysis and word segmentation.

Another problem is that large text classification datasets usually suffer from long tailed data distribution problem. This means that few classes make up majority of data. This problem often reduces the model's accuracy on the minority classes making more biased towards majority classes.

This problem can be addressed by either resampling (Chawla et al., 2002; Shen et al., 2016; Geifman and El-Yaniv, 2017; Buda et al., 2018; Zou et al., 2018) or re-weighting the cost function (Ting, 2000; Zhou and Liu, 2005; Huang et al., 2016; Khan et al., 2017; Cui et al., 2019).

Cui et al. (2019) noticed that re-weighting the cost function by inverse class frequency as used in vanilla schemes (Huang et al., 2016, 2019; Wang et al., 2017) could lead to poor performance on majority classes. They proposed class-balanced (CB) loss based on the effective number of classes which re-weights the loss by the inverse of the effective number of classes.

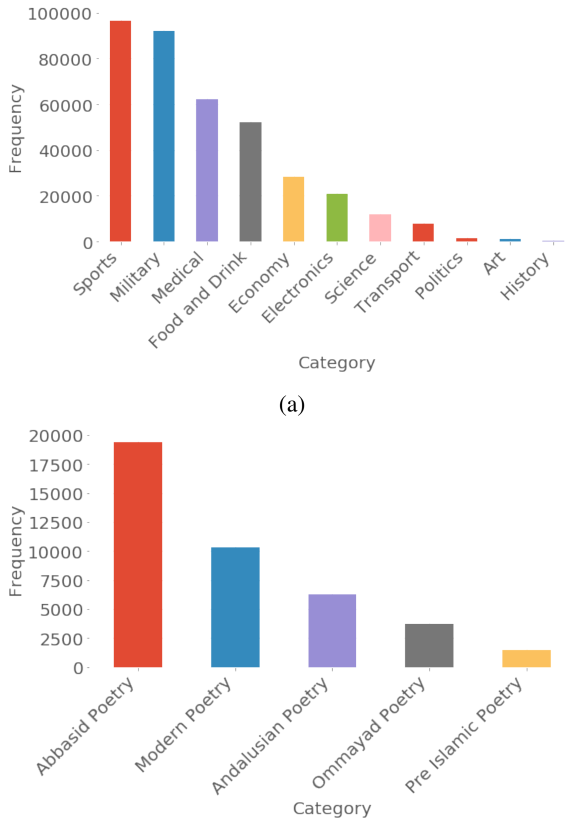
Our contributions can be summarized as follows:

- We propose AraDIC which is a framework for Arabic text classification. AraDIC is an end-to-end model of a character encoder and a classifier trained using CB loss.

- CB loss was originally intended for object detection tasks. We show that it can solve class-imbalance problems for text classification tasks.

- We introduce two datasets in the hope of becoming bench marking datasets for Arabic text classification tasks as well. The Arabic Wikipedia title (AWT) dataset and the the Arabic poetry (AraP) dataset. These two datasets contain the three types of Arabic language.

To the best of our knowledge, this is the first time an image-based character embedding model is used for Arabic text classification. Also, the first time a deep-learning based model is tested



(a)

(b)

Figure 1: The category distribution for the (a) AWT and (b) AraP datasets.

on datasets containing the three types of Arabic. This shows that our method could be used to overcome Arabic's complicated morphological analysis and word segmentation for all types of Arabic. The code and datasets are released at https://github.com/mahmouddaif/AraDIC

2 Datasets

Arabic text classification lacks bench marking datasets. This is because it is expensive and time consuming to annotate a large dataset to be used for text classification using deep learning algorithms. We created two large datasets that do not require manual annotation and can be used as benchmarks for Arabic text classification. The AWT and the AraP datasets. Sections 2.1 and 2.2 describe how we constructed these datasets.

2.1 Arabic Wikipedia Title Dataset (AWT)

Liu et al. (2017) introduced the Wikipedia title dataset for Japanese, Chinese and Korean by making use of Wikipedia's recursive hierarchical structure to crawl 12 different Wikipedia categories and using the category as a label to all article titles under this category, and its subcategories. He assumed that an article only exists in one category. If an article existed in more that one category, it was randomly assigned to only one of them. This created some noisy annotations, however, categories were chosen as distinctive in nature as possible to reduce this problem. We crawl 11 different categories from the Arabic Wikipedia using the same method. A total of 444,911 different titles with a total of 4,196,127 different words were crawled. This dataset contains mostly modern standard Arabic. The dataset category distribution can be found in Figure 1a.

2.2 Arabic Poetry Dataset (AraP)

The AraP dataset was crawled from the Adab Website[2] It contains Arabic poetry from the 6th to 21st centuries and consists of 41,264 poems from five eras. This dataset contains mostly colloquial and old Arabic. AraP's Category distribution details can be found in Figure 1b.

3 Methodology

AraDIC is an end-to-end framework of a character encoder (CE) and a classifier. We choose two classifiers for our framework. A character CNN (CLCNN) similar to Kitada et al. (2018), but tuned to Arabic language, and a bidirectional gated recurrent unit (BiGRU) (Chung et al., 2014) based classifier. The outline of our framework is shown in Figure 2. We use wildcard training introduced by (Shimada et al., 2016) for data augmentation. Wildcard training is dropout on the embedding space so that the data changes a little every training iteration. In that sense it acts as a data augmentation technique. We use CB softmax loss to deal with class imbalance problem.

3.1 Character Encoder

The CE is a CNN where convolution is performed in a depth-wise manner. It learns to encode each input character image of size 36 × 36 pixels into a 128-dimension vector. The architectural configuration is shown in Table 1a.

3.2 Classifier

For classification we use two classifiers. The first one is a CLCNN, and the second is a BiGRU. Input

[2] Adab website for Arabic poetry from 6th to 21st centuries. http://www.adab.com/.

Layer	Configuration
Conv2D	(c= 1, k = 3x3, f=32) + ReLU
Max-Pool2D	(k=2x2)
Conv2D	(c=32, k = 3x3, f=32) + ReLU
Max-Pool2D	(k=2x2)
Conv2D	(c=32, k = 3x3, f=32) + ReLU
FC	(800,128) + ReLU
FC	(128,128) + ReLU

(a) Character encoder architecture.

Layer	Configuration
Conv1D	(c= 128, k = 3, f=512) + ReLU
Max-Pool1D	(k=3)
Conv1D	(c=512, k=3, f=512) + ReLU
Max-Pool1D	(k=3)
Conv1D	(c=512, k = 3, f=512) + ReLU
Conv1D	(c=512, k = 3, f=512) + ReLU
FC	(1024,1024) + ReLU
FC	(1024,nc) + ReLU

(b) CLCNN architecture.

Layer	Configuration
BiGRU	(input = 128, hidden = 128, layer = 3) + BN
FC	(256,nc)

(c) BiGRU architecture.

Table 1: AraDIC's architectural configuration, **c** is input channels, **k** is kernel size, **f** is feature maps, **nc** is number of classes and **BN** is Batch Normalization (Ioffe and Szegedy, 2015).

text is represented as an array of character images each encoded into a 128 dimension vector using the CE. Those character embeddings are the input features for both the CLCNN and the BiGRU.

The CLCNN is a character-level CNN whose architectural details can be found in Table 1b.

The BiGRU takes those characters embeddings and computes a sentence level embedding. The sentence embedding is the average of all the hidden layers outputs of the BiGRU. These sentence level features are then passed to a fully connected layer followed by a softmax for class prediction. Detailed architecture of the BiGRU can be found in Table 1c.

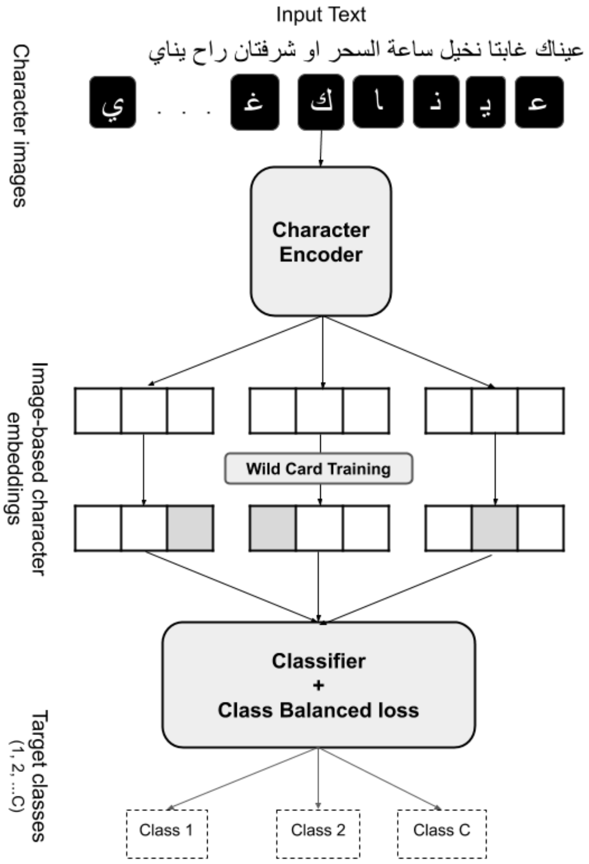

Input Text
عيناك غابتا نخيل ساعة السحر او شرفتان راح يناي
Character images
ع
ي
ن
ا
ك
غ
. . .
ي
Character Encoder
Image-based character embeddings
Wild Card Training
Classifier + Class Balanced loss
Target classes (1, 2...C)
Class 1
Class 2
Class C

Figure 2: AraDIC's architecture outline.

3.3 Class-Balanced Loss

Both of our datasets suffer from the long tailed distribution problem as shown in Figure 1a and 1b. To deal with this problem, we use state-of-the-art method, the class balanced loss (Cui et al., 2019). The class-balanced loss could be applied by reweighting the loss function by the inverse effective number of classes. We apply it to softmax cross entropy loss as follows:

$$-\frac{1-\beta}{1-\beta^{n_y}} \log\left(\frac{\exp\left(Z_y\right)}{\sum_{j=1}^{C}\exp\left(Z_j\right)}\right), \qquad (1)$$

where $\frac{1-\beta}{1-\beta^{n_y}}$ is the inverse effective number of classes. Z_j is the model output ($j = 1, 2, ...C$), y is class label for the input sample, n_y is number of samples per class y and β is a training hyper parameter. This will assign adaptive weights to the cost function for classes with higher samples and classes with lower samples, effectively re-weighting the cost function based on effective number of classes. This method was originally intended for object detection, we show that it can be applied to text classification as well.

Model			F-score			
			Arabic Wikipedia Title		Arabic Poetry	
	Embedding	Classifier	Micro [%]	Macro [%]	Micro [%]	Macro [%]
		Majority Class	21.67	2.97	47.06	5.33
Word level	Unigram	SVM	45.47	26.60	52.80	34.83
	AraVec	CNN	45.02	25.05	69.28	41.95
Character level	One-hot	CLCNN	42.76	18.71	68.24	37.72
	AraDIC	CLCNN (− CB loss)	47.47	26.85	74.86	45.61
		CLCNN (+ CB loss)	49.49	30.55	74.03	48.65
		BiGRU (− CB loss)	55.71	39.04	78.93	59.88
		BiGRU (+ CB loss)	**57.76**	**44.54**	**79.53**	**65.00**

Table 2: Classification results of our model and other baselines. **Majority Class**: Due to high class-imbalance in both of our datasets, we examine the performance of majority class classifier. **CNN + AraVec**: Sentence classifier CNN (Sagheer and Sukkar, 2018; Kim, 2014) using AraVec's word embeddings (Soliman et al., 2017). **SVM**: an SVM with unigrams, stemming, and document term matrix with TF-IDF scores as features. **CLCNN**: character level CNN with one hot encoding as inputs(Zhang et al., 2015). **AraDIC**: our proposed end-to-end framework of character encoder, CLCNN and BiGRU classifiers, trained with and without class-balanced softmax loss **(CB loss)**. We report two evaluation metrics, the macro and micro F-scores.

4 Experiments

To train our classifier both datasets are divided into 80% training data and 20% testing data[3].

4.1 AraDIC

The maximum character length or each document is set to 60 characters for the AWT dataset and 128 characters for the AraP dataset. That's for using the CLCNN classifiers. As for the BiGRU classifier we don't set a maximum character length, instead the whole text is used. Each character was encoded into a 128 dimension vector using the CE. Adam optimizer (Kingma and Ba, 2014) with a batch size of 64 and a learning rate of 0.001 was chosen as the optimization method. As for the CB loss we set β to 0.99 for both datasets. Wildcard training ratio is set to 10%. The training loss converged after approximately 150 epochs for AraP dataset and 500 epochs for AWT dataset.

4.2 Baselines

We use several word-based and character-based baselines to evaluate our method. They include both classical and deep learning baselines as follows:

- Due to high class imbalance in both our datasets, a majority class classifier is chosen as our first baseline.

- A classical Support Vector Machine (SVM) with a document-term matrix (DTM) of TF-IDF scores for unigrams as input was used as word-based baseline. Terms occurring only once and terms appearing in more than 90% of documents were omitted from the DTM. We performed preprocessing in the form of stop words, non-Arabic characters, diacritics removal. Then, text is stemmed using Khoja stemmer (Khoja, 2001). Farasa segmenter (Abdelali et al., 2016) was used for word segmentation.

- We also used Sagheer and Sukkar (2018)'s method of using AraVec's word embeddings as input features and sentence CNN originally introduced by Kim (2014) for classification. This is another word-based baseline.

- Another baseline is a character-level CNN (CLCNN) introduced by Zhang et al. (2015). In this baseline, input characters were one-hot encoded.

[3] Hyperparameters were tuned with a validation set split from the training set, and reported the predicted results of the evaluation set.

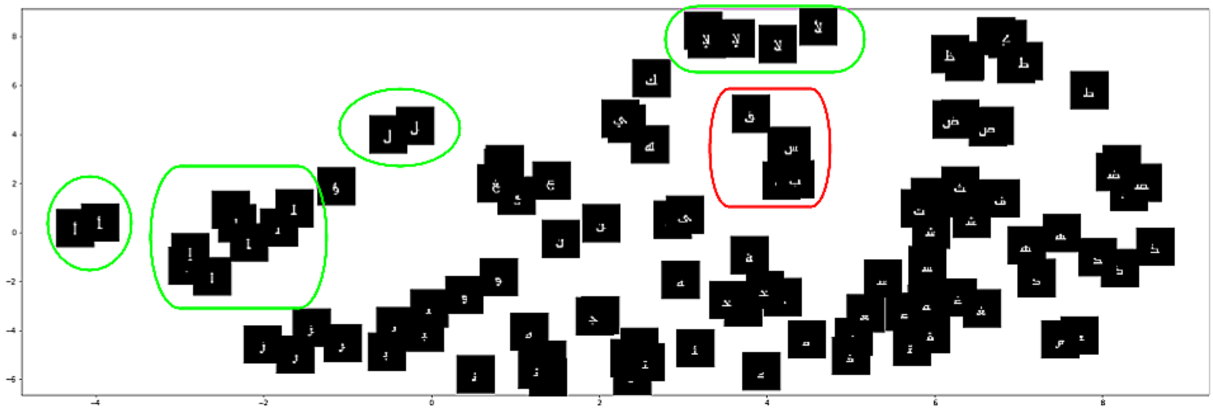


Figure 3: Character embeddings visualization using t-SNE (Maaten and Hinton, 2008). Sections circled in green show clusters of related characters with similar shapes, which was the majority of cases. Sections encircled in red show clusters of unrelated characters which was rare.

5 Results and Discussion

Classification results can be found in Table 2. It is noted that AraDIC outperforms both word based and character based deep learning and classical baselines. Performance improvement is shown over classical SVM without the need for preprocessing, word segmentation, stemming and feature engineering associated with classical methods. It was also able to beat Sagheer and Sukkar (2018) method of using sentence CNN with AraVec's word embeddings as input features without the need for word segmentation. This makes character level representations a better choice for Arabic language avoiding segmentation and feature engineering problems. It's also shown that using AraDIC's image-based character embeddings outperforms CLCNN with one-hot encoded characters as input features. Therefore, we can conclude as well that image-based character embeddings are useful for Arabic language due to the property of the language as discussed in the introduction section of this paper.

As for the classifier part of AraDIC, it can be noticed that the BiGRU significantly outperforms CLCNN for both classification tasks. This suggests that sequence-to-sequence models are more suitable for text classification using image-based character-based embeddings, especially in Arabic document classification.

Also, using CB loss improves the macro F-score of classifiers for both datasets. It can be also noted that the improvement in the macro F-score is achieved when using a CLCNN and a BiGRU. This shows that CB loss can be useful to solve class imbalance problems for text classification tasks.

Figure 3 shows character embeddings visualization using t-distributed stochastic neighbor embedding (t-SNE) method (Maaten and Hinton, 2008). As shown, embedding for related characters having similar shapes like “ أ ”, “ ا ”, “ ا ”, and “ إ ” are clustered in the embedding space. This is the majority of cases. Other unrelated characters are also clustered which is rare. This however shows that using image based character embeddings gives an extra layer of visual information. Another reason why it is useful is because both the CE and the classifier are trained as an end-to-end system. This means that the CE learns the best embeddings suitable for the classifier.

6 Conclusion

In this paper, we proposed a novel end-to-end Arabic text classification framework AraDIC. We also published two large scale Arabic text classification datasets that contain the three types of Arabic language, the AWT and the AraP datasets. AraDIC's image-based character embedding strategy eliminated the need for complicated preprocessing, segmentation and morphological analysis, and achieved much better performance than conventional deep and classical text classification techniques that use word and character-based embeddings. We have shown also that class-balanced loss is useful for text classification tasks with long tailed distribution datasets.

References

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for arabic. In *Proc. of NAACL*, pages 11–16.

Mohammed N Al-Kabi, Saif A Kazakzeh, Belal M Abu Ata, Saif A Al-Rababah, and Izzat M Alsmadi. 2015. A novel root based arabic stemmer. *Journal of King Saud University-Computer and Information Sciences*, 27(2):94–103.

Nora Al-Twairesh, Hend Al-Khalifa, and AbdulMalik Al-Salman. 2016. Arasenti: Large-scale twitter-specific arabic sentiment lexicons. In *Proc. of ACL*, pages 697–705.

Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proc. of NIPS Workshop on Deep Learning*.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-balanced loss based on effective number of samples. In *Proc. of CVPR*, pages 9268–9277.

Alaa M El-Halees. 2009. Filtering spam e-mail from mixed arabic and english messages: A comparison of machine learning techniques. *Filtering Spam E-Mail from Mixed Arabic and English Messages: A Comparison of Machine Learning Techniques.*, 6(1).

Yonatan Geifman and Ran El-Yaniv. 2017. Deep active learning over the long tail. *CoRR preprint arXiv:1711.00941*.

Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. 2016. Learning deep representation for imbalanced classification. In *Proc. of CVPR*, pages 5375–5384.

Chen Huang, Yining Li, Change Loy Chen, and Xiaoou Tang. 2019. Deep imbalanced learning for face recognition and attribute prediction. *IEEE transactions on pattern analysis and machine intelligence*.

Hossam S Ibrahim, Sherif M Abdou, and Mervat Gheith. 2015. Sentiment analysis for modern standard arabic and colloquial. *CoRR preprint arXiv:1505.03105*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. 2017. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 29(8):3573–3587.

Shereen Khoja. 2001. Apt: Arabic part-of-speech tagger. In *Proc. of the Student Workshop at NAACL*, pages 20–25.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR preprint arXiv:1412.6980*.

Shunsuke Kitada, Ryunosuke Kotani, and Hitoshi Iyatomi. 2018. End-to-end text classification via image-based embedding using character-level networks. In *Proc. of IEEE AIPR Workshop*, pages 1–4. IEEE.

Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. 2017. Learning character-level compositionality with visual features. In *Proc. of ACL*, pages 2059–2068.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. 2011. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pages 52–59. Springer.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, pages 3111–3119.

Fuchun Peng, Xiangji Huang, Dale Schuurmans, and Shaojun Wang. 2003. Text classification in asian languages without word segmentation. In *Proc. IRAL workshop*, pages 41–48.

Dania Sagheer and Fadel Sukkar. 2018. Arabic sentences classification via deep learning. *International Journal of Computer Applications*, 182(5):40–46.

Said A Salloum, Ahmad Qasim AlHamad, Mostafa Al-Emran, and Khaled Shaalan. 2018. A survey of arabic text mining. In *Intelligent Natural Language Processing: Trends and Applications*, pages 417–431. Springer.

Mohammed A Shehab, Omar Badarneh, Mahmoud Al-Ayyoub, and Yaser Jararweh. 2016. A supervised approach for multi-label classification of arabic news articles. In *Proc. of CSIT*, pages 1–6. IEEE.

Li Shen, Zhouchen Lin, and Qingming Huang. 2016. Relay backpropagation for effective learning of deep convolutional neural networks. In *Proc. of ECCV*, pages 467–482. Springer.

Daiki Shimada, Ryunosuke Kotani, and Hitoshi Iyatomi. 2016. Document classification through image-based character embedding and wildcard training. In *Proc. of IEEE Big Data*, pages 3922–3927. IEEE.

Abu Bakr Soliman, Kareem Eissa, and Samhaa R El-Beltagy. 2017. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117:256–265.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Kai Ming Ting. 2000. A comparative study of cost-sensitive boosting algorithms. In *Proc. of ICML*. Citeseer.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. 2017. Learning to model the tail. In *Advances in Neural Information Processing Systems*, pages 7029–7039.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proc. of NIPS*, pages 649–657.

Zhi-Hua Zhou and Xu-Ying Liu. 2005. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on knowledge and data engineering*, 18(1):63–77.

Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. 2018. Domain adaptation for semantic segmentation via class-balanced self-training. *arXiv preprint arXiv:1810.07911*.