The Journal of Electronic Theses and Dissertations

Volume 1 Article 6

2021

Otrouha: A Corpus of Arabic ETDs and a Framework for Automatic Subject Classification

Eman Abdelrahman
Virgnia Tech, Blacksburg, emanh@vt.edu

Fatimah Alotaibi
Virginia Tech, Blacksburg, falotaibi@vt.edu

Edward A. Fox
Virginia Tech, Blacksburg, fox@vt.edu

Osman Balci
Virginia Tech, Blacksburg, balci@vt.edu

Otrouha: A Corpus of Arabic ETDs and a Framework for Automatic Subject Classification

Eman Abdelrahman[1], Fatimah Alotaibi[2], Edward A. Fox[3], and Osman Balci[4]

[1]Department of Computer Science, Virginia Tech, emanh@vt.edu, ORCiD: 0000-0002-5298-8888

[2]Department of Computer Science, Virginia Tech, falotaibi@vt.edu, ORCiD: 0000-0001-9871-8966

[3]Department of Computer Science, Virginia Tech, fox@vt.edu, ORCiD: 0000-0003-1447-6870

[4]Department of Computer Science, Virginia Tech, balci@vt.edu, ORCiD: 0000-0002-2965-3035

Abstract

Although the Arabic language is spoken by more than 300 million people and is one of the six official languages of the United Nations (UN), there has been less research done on Arabic text data (compared to English) in the realm of machine learning, especially in text classification. In the past decade, Arabic data such as news, tweets, etc. have begun to receive some attention. In contrast, Arabic **E**lectronic **T**heses and **D**issertations (ETDs) have received little attention, in spite of the huge number of benefits they provide to students, universities, and future generations of scholars. There are two main roadblocks to performing automatic subject classification on Arabic ETDs, which could be helpful for discovery and browsing. The first is the unavailability of a public corpus of Arabic ETDs. The second is the linguistic complexity of the Arabic language; that complexity is particularly evident in academic documents such as ETDs. To address these roadblocks, this paper presents Otrouha, a framework for automatic subject classification of Arabic ETDs, which has two main goals. The first is building a Corpus of Arabic ETDs and their key metadata such as abstracts, keywords, and title, to pave the way for more exploratory research on this valuable genre. The second is to provide a framework for automatic subject classification of Arabic ETDs through different classification models that use classical machine learning as well as deep learning techniques. The first goal is aided by searching the AskZad Digital Library, which is part of the Saudi Digital Library (SDL). AskZad provides other key metadata of Arabic ETDs, such as abstract, title, and keywords. The current search results consist of abstracts of Arabic ETDs. This raw data then undergoes a pre-processing phase that includes stop word removal using the Natural Language Tool Kit (NLTK), and word lemmatization using the Farasa API. To date, abstracts of 518 ETDs across 12 subjects have been collected. For the second goal, the preliminary results show that among the machine learning models, binary classification (one-vs.-all) performed better than multiclass classification. The maximum per subject accuracy is 95%, with an average accuracy of 68% across all subjects. It is noteworthy that the binary classification model performed better for some categories than others. For example, *Applied Science and Technology* shows 95% accuracy, while the category of *Administration* shows 36%. Deep learning models resulted in higher accuracy but lower F-measure; their overall performance is lower than machine learning models. This may be due to the small size of the dataset as well as the imbalance in the number of documents per category. Work to collect additional ETDs will be aided by collaborative contributions of data from additional sources.

Keywords: Arabic ETDs, Classification, Digital Libraries, Information Retrieval

Otrouha: A Corpus of Arabic ETDs and a Framework for Automatic Subject Classification

Introduction

Text classification is one of the major research topics in machine learning and Natural Language Processing (NLP). It is the process of categorizing and labeling text, based on its content, into predefined classes or categories. In the digital age, the amount of textual data available online is growing rapidly, which makes text classification essential to efficiently access, browse, and preserve that data.

While some Arabic text data has been studied in recent years, most of that consists of news articles and tweets, which are readily available to the general populace. However, **E**lectronic **T**heses and **D**issertations (ETDs) have received less attention, even though they provide a number of benefits to students, researchers of all backgrounds, scholars, and universities.

This is mainly due to two major roadblocks: (1) the unavailability of a large public corpus of Arabic ETDs and (2) the linguistic complexity of the Arabic language. The latter places a heavy strain on the pre-processing of the data and creates a challenge to apply common machine learning and deep learning techniques effectively.

In this paper we present Otrouha. Its first goal is to collect metadata for Arabic ETDs by searching the AskZad Digital Library, which is part of the Saudi Digital Library (SDL), to start a corpus of Arabic ETDs. This will pave the way for more research to take place on Arabic ETDs.

The second goal of this project is to provide a framework for automatic subject classification of Arabic ETDs. They are an unexplored genre of data in the realm of machine learning. Through subject classification, the Otrouha project can help to organize and manage Arabic ETDs in digital libraries.

Our dataset consists of some key metadata of Arabic ETDs such as abstracts. Subsequent pre-processing included stop word removal using NLTK, and word lemmatization using the Farasa API.

In recent years, various machine learning techniques have been employed to enable text classification, starting with traditional supervised-learning models, such as *k*-**N**earest **N**eighbor (*k*-NN) and **S**upport **Ve**ctor **M**achine (SVM) (Ahmed & Elhassan, 2015). More challenging combinations of models use deep learning algorithms, such as **C**onvolutional **N**eural **N**etworks (CNN) and **F**ully-**C**onnected (FC) neural networks. These methods are yielding increasingly robust and efficient results; however, such developments are almost solely based on languages such as English and Chinese (El-Shishtawy & El-Ghannam, 2014). Semitic languages such as Arabic, Amharic, Tigrinya, Hebrew, and various Middle-Eastern dialects have a very distinct construction and a complex form with inflections and derivations that make it harder to build highly-accurate automatic text classification models. Therefore, one of Otrouha's goals is to apply these techniques to a previously untapped source of data (Arabic ETDs) and evaluate their performance.

The remainder of the paper is ordered as follows. The Related Work section describes studies on text classification. The Approach section has subsections covering work on the Data and Classification. The former introduces our dataset, while the latter presents the classification techniques and models. The Results section gives numerical values for accuracy, F-measure, precision, and recall – based on our experimentation. Lastly, the Insights and Future Work section discusses the problem of imbalanced data, as well as our plans for additional research.

Related Work

Arabic text classification is a very challenging research area. Most previous research on text classification has been conducted on English datasets; the research on Arabic datasets is extremely limited. To the best of our knowledge, there has been no research on automatic classification of Arabic ETDs. In this section, we present related work in the Arabic text classification area. Research that has used machine learning and deep learning for Arabic text classification has helped us to make a baseline for our experiments.

Classical Machine Learning

Different classification models have been used in research on Arabic text classification, such as decision trees, Naïve Bayes, and support vector machines. Also, the pre-processing techniques vary. For example, in (Duwairi, 2007), pre-processing techniques include stop word removal, root extraction, and stemming for dimensionality reduction. A performance comparison has been made between different classification techniques such as Naïve Bayes, *k*-NN, and distance–based classification methods, using an Arabic dataset of 1,000 documents. To compare the accuracy between these classifiers, the authors used error rate, recall, precision, and fallout. The experiment shows that Naïve Bayes outperforms the *k*-NN and distance–based methods.

In (Al-Thubaity et al., 2008), an automated tool for Arabic text classification (ATC) has been presented. One goal of this paper was building a representative training dataset that covers different types of text categories, which can be used for further research. To this end, they used seven different datasets that contained 17,658 texts with more than 11,500,000 words as their corpus. The second goal was making a performance comparison between the SVM and C5.0 algorithms on the datasets. In general, the study found that the C5.0 algorithm outperformed the SVM algorithm by approximately 10%.

The work in (Gharib et al., 2009) includes using SVM, Naïve Bayes, *k*-NN, and Rocchio classifiers to categorize Arabic text. In order to test the classifiers, they conducted two experiments. In the first one, they used the training set as the test set, while the second experiment involved the leave-one testing method. The study found that the Rocchio classifier gave better results when the size of the feature set is small, while SVM outperformed the other classifiers when the size of the feature set is large.

The behavior of n-gram frequency statistics for Arabic text classification was studied in (Khreisat, 2006). The author employed the Manhattan distance for dissimilarity measure and Dice's measure of similarity along with an n-gram frequency statistics technique. The experiment showed better results for n-gram frequency with the Dice rather than Manhattan measure.

An Arabic document categorization tool was developed in (Kourdi et al., 2004) using the Naïve Bayes algorithm. For this study, non-vocalized Arabic web documents were classified according to five predefined categories using 300 web documents. A cross-validation experiment using 2,000 terms/roots showed an average accuracy over all categories of 68.78%, where the best categorization performance per category was 92.8%.

In contrast to the previous literature, (Sawaf et al., 2001) conducted an experiment on a large Arabic NEWSWIRE corpus without pre-processing. The authors posited that statistical methods are very powerful techniques for Arabic text classification and clustering (maximum entropy). The results show 89.5%, 31.5%, and 46.61% for recall, precision, and F-measure, respectively, which is an adequate result, given the lack of morphological analysis.

As stated in (El-Halees, 2007), developing a classifier for Arabic text is difficult due to the complexity of Arabic morphological analysis. The Arabic language has high inflectional and derivational morphology, which makes NLP tasks nontrivial. In this research, they used the maximum entropy framework to build a system (ArabCat) that works as a classifier for Arabic documents. Their results show that ArabCat gave 80.48%, 80.34%, and 80.41% for recall, precision, and F-measure, respectively, while Sakhr's categorizer shows 73.78%, 47.35%, and 57.68%, respectively.

Deep Learning

In the area of deep learning, there are a number of research studies on Arabic text classification. For example, (Harrag & Al-Qawasma, 2010) pre-processed their data by transforming all text into a term-document matrix (T-D matrix) before remodeling it based on **S**ingular-**V**alue **D**ecomposition (SVD) to reduce its dimensionality. The latter became the input for a three layer feed-forward neural network with hyperbolic tangent (tanh) activation function in the hidden layer, followed by a linear output layer. In their evaluation, they presented best average recall values of 50-53% and best average precision values of 53-55%.

Alternatively, (Jindal, 2016)'s study consists of using a combination of Markov clustering with a deep learning approach where documents are pre-processed by tokenizing them, segmenting them into different words, and extracting the TF-IDF weighted root-word counts (which are used as features). Clustering is then performed on the features using fuzzy-c-means and the Markov model, before training a deep belief network for each cluster using restricted Boltzmann machines, which the author claims improves classification accuracy and feature extraction. The evaluation stage included a 10-fold cross validation on 12,000 news documents which resulted in an F-measure of 91.02%.

(Boukil et al., 2018) employed a convolutional neural network (CNN) trained on Arabic text found online. They used a web-crawler to build a 319 million Arabic words corpus, applied the TF-IDF technique, and trained the CNN model using stochastic gradient descent (backpropagation), a learning rate of 0.001, and Keras' dropout function to enable the model to better converge. They found that two important features in classifying Arabic words were filter sizes and feature maps. The highest accuracy was 92.94% with 111,000 words.

(Dahou et al., 2019) involves a convolutional neural network combined with a differential evolution algorithm to perform sentiment classification on Arabic text. Their contributions include building and training different CNN architectures with variable numbers of parallel convolution layers, integrating two different mutation strategies to improve the exploration and exploitation abilities of the differential evolution algorithm, and using two different fitness evaluation techniques to assess the generalization of the CNN.

None of these research efforts have focused on Arabic ETDs, which is the core contribution of our research. Instead, the majority of prior research has been applied to other forms of Arabic text such as news and tweets. Furthermore, most of this classification work has relied on stemming the Arabic words in their pre-processing phase, whereas lemmatization has been performed in our work.

Approach

Since there is no available corpus of Arabic ETDs to be directly used, one of this project's goals is to build a seed corpus by collecting abstracts and other key metadata of Arabic ETDs. This will encourage more research to be done in this area.

The ETDs collected underwent a pre-processing phase to remove noisy and unwanted data. A classification model was then built and trained on 70% of the data before testing on the remaining 30%. Figure 1 shows the workflow of the Otrouha framework.

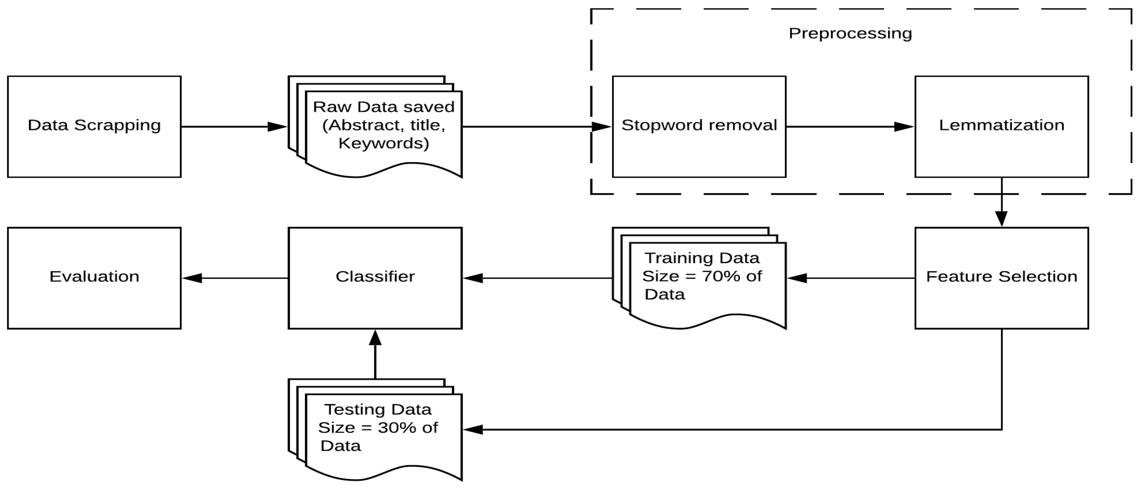
Preprocessing
Data Scrapping
Raw Data saved (Abstract, title, Keywords)
Stopword removal
Lemmatization
Evaluation
Classifier
Training Data Size = 70% of Data
Feature Selection
Testing Data Size = 30% of Data

Figure 1

The workflow of the Otrouha framework

Data

Data Collection

After a thorough exploration, we found the Saudi Digital Library (SDL) to contain several databases such as ProQuest, AskZad, Saudi Cultural Mission in Australia, Dar Almandumah, etc. We chose AskZad as our case study for the following reasons:

- It is rich in Arabic ETDs. According to (Arabia Inform, 2020), AskZad is considered as the premier place for Arabic academic research. It contains direct image scans of original articles going as far back as 1823.

- In addition to scanned ETDs, it provides abstracts and associated bibliographies for information enhancement, which would facilitate the data collection for the first phase of this project.

- Its categorization system is consistent with categorization systems of other digital libraries, such as ProQuest.

To collect the data from the AskZad digital library, we designed and developed a web crawler to perform the data collection process. Our preliminary work has led to a dataset of 518 records that we have manually validated. Since our focus is on classification, we postponed further work with the crawler once we had a small collection with roughly balanced coverage across a reasonable number of categories.

Later we will improve the crawler to use more of the AskZad records. That will take some time since we must limit the requests per second our crawler can make to the site. Our crawler will need a more robust and flexible parser of the pages served from AskZad, so we can accurately extract at least the most important metadata, e.g., abstract, title, and keywords.

Categories and Data Exploration

We analyzed the AskZad digital library to gain a deeper understanding of their categorization system. It has 16 categories, where the number of ETDs in each category varies. For example, the *Education* category has about 6,000 ETDs, whereas the *Culture* category has about 59 ETDs. Each of the ETDs is given one of the 16 categories. Table 1 shows the distribution of the data collected. Our preliminary collection covers a subset of that, i.e., 12 categories, since the number of ETDs in each of the other categories is small.

Mapping of AskZad to ProQuest Categories

As mentioned earlier, one of the reasons that made AskZad digital library the most suitable source of data for our project is its taxonomy. This taxonomy is found to be simple and consistent with ProQuest's taxonomy, which has the largest commercial multidisciplinary full-text ETD database.

Moreover, the AskZad digital library provides structured and consistent metadata, along with the scanned full-text PDF of every Arabic ETD. This consistency has been assured since the language used with the AskZad website can be English, Arabic, or French. Therefore, we used the English version of the AzkZad website to get the translation of each category and did a manual mapping for the translated AskZad categories to the ProQuest categories for 2018-2019, as shown in Figure 2.

For example, the *Law* category in AskZad is mapped to the high-level category *Law and Legal Studies* and its sub-categories in ProQuest. All categories in AskZad have been mapped to one or more corresponding categories in ProQuest, which ensures future consistency in categorization among different digital libraries of Arabic ETDs.

Data Pre-processing

Our dataset consists of abstracts of ETDs across 12 categories. In order to clean the raw data, some pre-processing techniques were applied. This included stop word removal

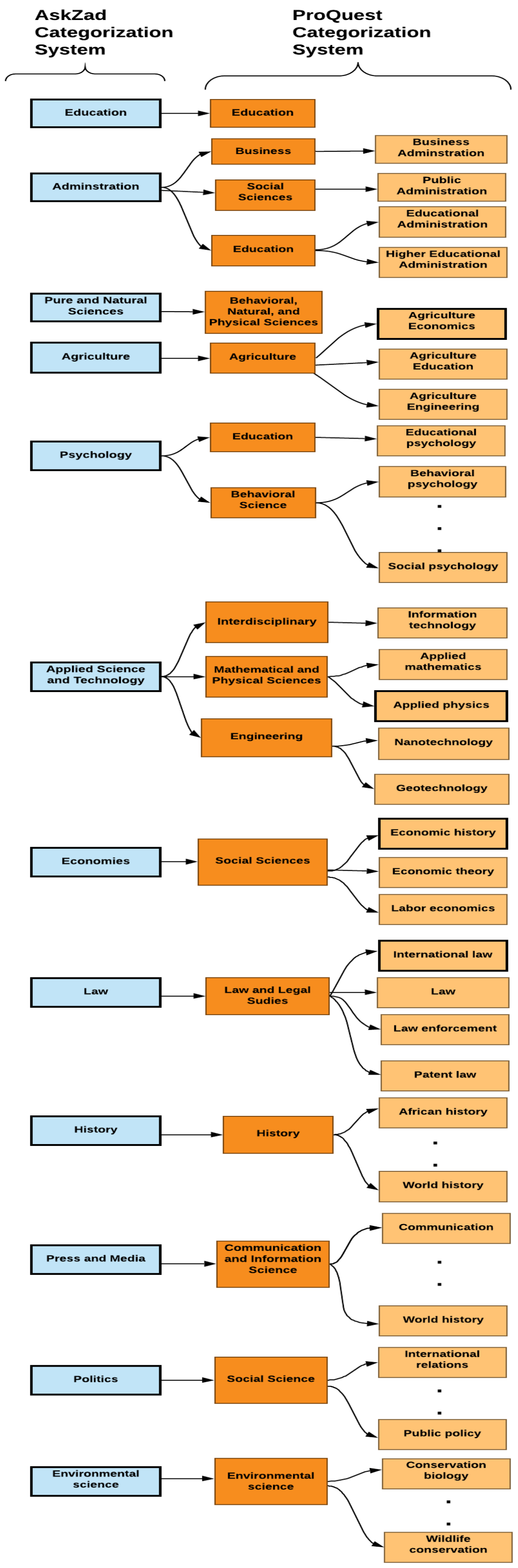

Figure 2

Mapping AskZad categories to ProQuest categories.

Table 1

Dataset Distribution Across The Categories

Category	Number of ETDs
Education	50
Administration	50
Pure & Natural Science	18
Agriculture	29
Psychology	47
Applied Science & Technology	11
Economics	37
Law	50
History	50
Press & Media	78
Politics	80
Environmental Science	18

and lemmatization.

a) Stop Word Removal: We removed the stop words that are considered meaningless or are too frequent, by using a library provided by the Natural Language Tool Kit (NLTK). For example, the number of words in a raw abstract originally was 204 words, while after stop word removal, it contained 168 words. Thus, approximately 17% of the raw data was removed for a better classification.

b) Lemmatization Process: The Arabic language has a very rich morphology, as roots can be employed to derive a lot of words. For example, a root of a verb can have a prefix, infix, and suffix. Moreover, suffixes refer to both the time and the tense of the verb. Finally, the gender of the participant in the verb should be added to the verb whether it is singular or plural (Duwairi, 2007). This richness increases the dimensionality of word vectors. Therefore, we needed to get the root of each word in each abstract in the dataset. We studied the impact of stemming in **N**atural **L**anguage **P**rocessing (NLP) tasks for Arabic, and throughout our research we found that lemmatization shows more efficacy than stemming, particularly in text summarization (El-Shishtawy & El-Ghannam, 2014), text indexing (Hammouda & Almarimi, 2010), and text classification (Koulali et al., 2013). In the lemmatization process, vocabulary and morphological analysis are used to find the base form (dictionary form) of a word by considering its inflected forms. On the other hand, the stemmer tries to strip prefixes and suffixes from words to leave a valid stem (Mubarak, 2018), but this does not always yield a meaningful word in the Arabic language. Therefore, we adopted lemmatization instead of stemming. Different lemmatizers has been tested on a sample abstract to compare their performance. We found the Farasa lemmatizer (Abdelali et al., 2016) gave more accurate output and outperformed the state-of-the-art MADAMIRA (Pasha et al., 2014) and Stanford Arabic segmenter in regard to Arabic segmentation and lemmatization. As a result, we used the Farasa API to lemmatize words in each abstract. Farasa has a REST API that gets an HTTP POST request and responds by returning the lemmatized abstract. To ensure the POST request returns valid data, an **A**dvanced **R**EST **C**lient (ARC) tool[1] has been used for manual testing.

Classification

In order to provide automatic text classification that can improve the browsability and accessibility of Arabic ETDs, we tried both supervised machine learning as well as deep learning classification techniques. Collected data has been split into training and testing in the ratio of 70% and 30%, respectively.

[1] https://install.advancedrestclient.com/install

Supervised Machine Learning

According to previous work done in the area of text classification, the following algorithms are commonly used: **S**upport **V**ector **M**achines (SVMs), **R**andom **F**orest (RF), and **D**ecision **T**rees (DT).

The feature set of each abstract was extracted using "CountVectorizer" from the scikit-learn library. According to (Buitinck et al., 2013), scikit-learn is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data pre-processing, model selection, evaluation, and many other utilities. CountVectorizer converts a collection of text documents to a matrix of token counts. Since research in this area is scarce, we conducted both binary and multiclass classifications to compare the performance of these algorithms for the Arabic ETDs classification task.

Multiclass Classification. First, we conducted multiclass classification. Given a classification problem with N classes, where N is more than two, a text is assigned to one of the N classes. We tried Naïve Bayes, decision tree, random forest, and SVM classifiers. An ensemble classifier, consisting of random forest and support vector machines classifiers, has been trained to determine whether classification accuracy will be improved. The performance of each of the aforementioned classifiers was relatively poor, with an average accuracy of 24%.

Binary Classification. To leverage binary classification, where a text is assigned to either one of two classes, we used a one-vs.-all approach. Given a classification problem with N classes, a one-vs.-all approach consists of N separate binary classifiers, where one binary classifier for each possible class is used. Since among single classifiers the random forest classifier had the highest accuracy, we used it to perform binary classification. This greatly improved the classification performance, where the average accuracy was 68%.

Supervised Deep Learning

Since deep learning has been recently regaining attention in the text classification literature and giving promising results, we ran experiments using Keras. According to (Keras Project, 2020), Keras was the most used deep learning framework among the top-5 winning teams on Kaggle. Since this is exploratory research, we tried replicating different existing models as baselines to be able to see how each of them performed with our dataset.

CNN with Embedding Layer. The first model we tried (Falbel et al., 2019) was based on **C**onvolutional **N**eural **N**etworks (CNN). It worked well on the dataset of IMDB[2] which is the world's most popular and authoritative source for movie, TV, and celebrity content (Maas et al., 2011). This model pads the sequences before feeding the neural network. CNN modeling starts off with an embedding layer, which maps vocabulary indices into embedding dimensions. Then it adds a Convolution1D, which will learn filters, followed by a max pooling layer. After that, it projects onto a single unit output layer and compresses it with a sigmoid function. Then, the model is run using a binary cross entropy loss function and Adam optimizer. To prevent overfitting of the CNN, a dropout regularization technique was adopted with different rates that range from 0.2 to 0.9. Dropout was used in three positions: after each of the embedding layer, pooling layer, and **F**ully-**C**onnected (FC) layer. Also, the Adam optimizer was used to train the CNN. ReLU and sigmoid functions are used as activation functions for both convolution and output layers. This model resulted in an accuracy of 10% for our dataset, while for the IMDB dataset, it resulted in accuracy of 89% after 2 epochs. Therefore, we also tried the following model.

CNN without Embedding Layer. Another use of the Keras API has been to implement an intent classification module for Arabic text data using CNN on a Twitter dataset (Bashir, 2018). It differs in the conversion of data from the aforementioned model, since it does not use an embedding layer, but the CNN architecture was the same. Using

[2] https://developer.imdb.com/

this model on our dataset resulted in accuracy of approximately 12%.

Modified Model. Since the accuracy we achieved using existing models was a maximum of 12%, we tried several experiments and parameter tuning in order to determine what can be negatively affecting the performance. To convert the text abstracts into numeric values, we used the "texts_to_sequences" Keras function, which transforms each text into a sequence of integers. For the neural network architecture, we removed the convolutional layer and used only the fully-connected neural network. These resulted in achieving a great improvement in the accuracy, which increased to reach 87%. However, the F1, precision, and recall were all approximately 15%, which is very low. The potential reasons behind this are discussed in the following section.

Results

This section provides a detailed overview of the results of the experiments. It starts with the machine learning classification, including the multiclass classification as well as the binary classification. Then, results of deep learning experiments are presented.

Supervised Machine Learning

Multiclass Classification

The accuracies presented in Table 2 show that the SVM classifier gave 0.237, while the decision tree classifier gave an accuracy of 0.244, and the random forest classifier gave 0.252. An ensemble classifier consisting of the aforementioned classifiers and Gaussian Naïve Bayes was tried in order to determine how impactful combining the classifiers can be on the accuracy. The ensemble classifier raised the accuracy by only 0.7%, to give an overall accuracy of 0.259.

Binary Classification

Among the classifiers tried in multiclass classification, the random forest classifier was found to give the highest accuracy. For that reason, we chose it to conduct our second

Table 2

Results using Multiclass Classification

Classifier	Accuracy
Support Vector Machines (SVM)	0.237
Decision Trees	0.244
Random Forest (RF)	0.252
Ensemble Classifier (RF, SVM, and Gaussian Naïve Bayes)	0.259

set of experiments, which was the binary classification (one-vs.all). For our second experiments, we included more evaluation metrics such as precision, recall, and F-measure. The results in Table 3 show that binary classification performed much better than the multiclass classification. Among the 12 categories, eight of them achieved an accuracy of 50% or higher. Two of these eight categories yielded the highest accuracy as well as highest F-measure. These two categories are *Applied Science and Technology* and *Economics*, where the highest accuracies were 95% and 80%, respectively. The other six categories had satisfactory results for precision, recall, and F-measure. The remaining four categories resulted in less than 50% accuracy.

Supervised Deep Learning

The average results of our deep learning model using the fully-connected layers model are shown in Table 4. Our model includes three Keras dense layers using a **R**ectified **L**inear **U**nits (ReLU) activation function followed by a dropout layer and a final dense layer using a softmax activation layer. Using this architecture, we were able to achieve an accuracy value ranging between 85-88%. However, our F-measure, precision, and recall percentages remained very low, ranging between 10-20%. The insights gained from these experiments are discussed in the next section.

Table 3

Results using Binary Classification (One-vs.-all)

Category	Precision	Recall	F1	Accuracy
Education	0.69	0.66	0.65	0.66
Administration	0.18	0.45	0.26	0.36
Pure & Natural Science	0.62	0.63	0.62	0.63
Agriculture	0.69	0.65	0.64	0.66
Psychology	0.21	0.50	0.30	0.43
Applied Science & Technology	0.95	0.95	0.95	0.95
Economics	0.80	0.79	0.79	0.80
Law	0.57	0.57	0.53	0.53
History	0.43	0.43	0.39	0.49
Press & Media	0.42	0.42	0.42	0.46
Politics	0.30	0.37	0.37	0.60
Environmental Science	0.79	0.60	0.54	0.63

Insights and Future Work

Imbalanced Categories

Some additional research has been done on the interpretation of the shown results, specially the large gap between the accuracy values being quite high and the F-measure, precision, and recall being on the lower end. Research shows that such gap is due to the imbalance in the dataset, as shown in Table 1, which leads to skewness that degrades the model's performance (Koehrsen, 2018).

To further test this argument, we gave a closer look at the dataset distribution. Our dataset consists of 12 different classes of ETDs' metadata to categorize. The number of them per category is not equal; some categories contain 80 documents while others only

Table 4

Results using Deep Learning Techniques

Average Values/ Model	Accuracy	F1	Precision	Recall
Reference model (IMDB)	97.29	94.17	96.4	92.1
Our modified model	87.3	15.48	15.48	15.48

contain 11 documents. Therefore, we investigated which values were predicted most and found that the *Politics* class was the most commonly predicted one. This class contains the highest number of metadata records. We hypothesize that by having a larger and more balanced dataset, we would be able to achieve higher F-measure and continue improving on our accuracy values. The size of the dataset is expected to contribute to the robustness of the results since deep learning algorithm performance increases as the data size increases.

The findings also show that removing the convolutional layer positively affected the model's performance. Therefore, we plan on trying other architectures such as Long Short-Term Memory (LSTM) networks, which do well with sequences of data.

Future Work

The exploratory research, along with the experiments and results, gave us insights into how to proceed when moving forward. Greatly enlarging the corpus will be needed, since the larger the corpus gets, the more robust the results become. Gathering more data points is expected to improve each of the models' performance as well (Matykiewicz & Pestian, 2012) (Zhang et al., 2015).

Therefore, we plan to modify our web crawler to work better with the AskZad digital library website. Moreover, we can collect the corresponding English version of the data so that we can compare each of the model's performance on the same data but with a different language.

The experiments can then be rerun using the enlarged corpus to determine how much the model's performance is impacted by the size of the dataset. This will be of interest for researchers willing to work in the area of Arabic ETDs for further investigation and bench-marking in NLP.

In addition to exploring the effects of number of documents, we also plan to investigate the differences among the vocabularies associated with the different categories. For example, each of the *Education* and *Administration* categories has 50 documents. However, between these, the accuracy and F-measure values achieved vary greatly, i.e., 66% and 36% respectively. Therefore, in future work, we will use language models and confusion matrices to compare the categories and classification results. We aim to determine whether there is a pattern that can be detected in the mis-classification, so we can improve the classifiers accordingly.

In the deep learning model, our findings show that using text to sequence for word conversions negatively affects the context. Since Arabic is a highly complex language where words are highly dependent on the context, this in turn negatively affected the model's performance. In future work, we plan to use word embeddings by using the FastText Library, which has pre-trained word vectors of 157 languages, including Arabic. This is expected to improve the model's performance since it better preserves the context.

Acknowledgment

Thanks go to IMLS for providing support through grant LG-37-19-0078-19. Thanks also go to the AskZad Digital Library, which is part of the Saudi Digital Library, for providing access to data. We would also like to thank Palakh Jude and Yasmine Belghith for their contributions.

References

Abdelali, A., Darwish, K., Durrani, N., & Mubarak, H. (2016). Farasa: A Fast and Furious Segmenter for Arabic. *Proceedings of the 2016 conference of the North American chapter of the Association for Computational Linguistics*, 11–16.

Ahmed, M., & Elhassan, R. (2015). Arabic text classification review. *Journal of Computer Science and Software Engineering*, *4.1*, 1–5.

Al-Thubaity, A., Khorsheed, M., Al-Harbi, A., Almuhareb, A., & Al-Rajeh, A. (2008). Automatic Arabic text classification. *JADT 2008 : 9es Journées internationales d'Analyse statistique des Données Textuelles*. https://eprints.soton.ac.uk/272254/1/Arabic-Classification.pdf

Arabia Inform. (2020). AskZad, The World's First and Largest Arabic Digital Library [Accessed 14 June 2020]. http://askzad.com/

Bashir, A. (2018). Intent classification module for Arabic text data using CNN [Accessed 14 June 2020]. https://github.com/abdallah197/text_classification_CNN_arabic

Boukil, S., Biniz, M., El Adnani, F., Cherrat, L., & El Moutaouakkil, A. E. (2018). Arabic text classification using deep learning technics. *International Journal of Grid and Distributed Computing*, *11*(9), 103–114. https://www.researchgate.net/publication/327968032_Arabic_Text_Classification_Using_Deep_Learning_Technics

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). API design for machine learning software: Experiences from the scikit-learn project. *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 108–122.

Dahou, A., Elaziz, M. E. A., Zhou, J., & Xiong, S. (2019). Arabic sentiment classification using convolutional neural network and differential evolution algorithm. *Computational intelligence and neuroscience*, *2019*.

Duwairi, R. M. (2007). Arabic text categorization. *International Arab Journal of Information Technology*, *4*(2), 125–131.

El-Halees, A. M. (2007). Arabic text classification using maximum entropy. *IUG Journal of Natural Studies*, *15*, 157–167.

El-Shishtawy, T., & El-Ghannam, F. (2014). *A lemma based evaluator for Semitic language text summarization systems* [arXiv abs/1403.5596]. `https://arxiv.org/pdf/1403.5596.pdf`

Falbel, D., Allaire, J., & Chollet, F. (2019). Text classification using Convolution1D [Accessed 1 April 2021]. `https://keras.rstudio.com/articles/examples/imdb_cnn.html`

Gharib, T. F., Habib, M. B., & Fayed, Z. T. (2009). Arabic text classification using Support Vector Machines. *Int. J. Comput. Their Appl.*, *16*, 192–199.

Hammouda, F. K., & Almarimi, A. (2010). Heuristic lemmatization for Arabic texts indexation and classification. *Journal of Computer Science*, *6*, 660–665.

Harrag, F., & Al-Qawasma, E. (2010). Improving Arabic text categorization using neural network with SVD. *J. Digit. Inf. Manag.*, *8*, 233–239.

Jindal, V. (2016). A personalized Markov clustering and deep learning approach for Arabic text categorization. *Proceedings of the ACL 2016 Student Research Workshop*, 145–151.

Keras Project. (2020). Keras: The Python deep learning API [Accessed 14 June 2020]. `https://keras.io/`

Khreisat, L. (2006). Arabic text classification using N-gram frequency statistics: A comparative study. *DMIN*, *2006*, 78–82.

Koehrsen, W. (2018). Beyond accuracy: Precision and recall [Accessed 14 June 2020]. https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c

Koulali, R., El-Haj, M., & Meziane, A. (2013). Arabic topic detection using automatic text summarisation. *2013 ACS International Conference on Computer Systems and Applications (AICCSA)*, 1–4.

Kourdi, M. E., Bensaid, A., & Rachidi, T.-E. (2004). Automatic Arabic document categorization based on the Naïve Bayes algorithm. *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*, 51–58.

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. *Proceedings of the 49th annual meeting of the Association for Computational Linguistics: Human language technologies*, 142–150.

Matykiewicz, P., & Pestian, J. (2012). Effect of small sample size on text categorization with support vector machines. *BioNLP@HLT-NAACL*, 193–201.

Mubarak, H. (2018). *Build fast and accurate lemmatization for Arabic* [arXiv abs/1710.06700]. https://arxiv.org/pdf/1710.06700.pdf

Pasha, A., Al-Badrashiny, M., Diab, M. T., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O., & Roth, R. (2014). Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. *LREC*, *14*(2014), 1094–1101.

Sawaf, H., Zaplo, J., & Ney, H. (2001). Statistical classification methods for Arabic news articles. *Arabic Natural Language Processing in ACL2001.*

Zhang, X., Zhao, J. J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. *NIPS, Advances in neural information processing systems*, 649–657.