# Automatic Categorization of Islamic Jurisprudential Legal Questions using Hierarchical Deep Learning Text Classifier


| Wesam H. AlSabban | Saud S. Alotaibi | Abdullah Tarek Farag | Omar Essam Rakha | Ahmad A. Al Sallab | Majid Alotaibi |
|---|---|---|---|---|---|
| whsabban@uqu.edu.sa | ssotaibi@uqu.edu.sa | abdullahtarek57@gmail.com | i@omarito.me | ahmad.elsallab@gmail.com | mmgethami@uqu.edu.sa |
| Department of Information Systems, Umm Al-Qura University | Department of Information Systems, Umm Al-Qura University | Speakol | Faculty of Engineering, Ain Shams University | Faculty of Engineering, Cairo University | Department of Computer Engineering, Umm Al-Qura University |



**Summary**
The Islamic jurisprudential legal system represents an essential component of the Islamic religion, that governs many aspects of Muslims' daily lives. This creates many questions that require interpretations by qualified specialists, or Muftis according to the main sources of legislation in Islam. The Islamic jurisprudence is usually classified into branches, according to which the questions can be categorized and classified. Such categorization has many applications in automated question-answering systems, and in manual systems in routing the questions to a specialized Mufti to answer specific topics. In this work we tackle the problem of automatic categorisation of Islamic jurisprudential legal questions using deep learning techniques. In this paper, we build a hierarchical deep learning model that first extracts the question text features at two levels: word and sentence representation, followed by a text classifier that acts upon the question representation. To evaluate our model, we build and release the largest publicly available dataset of Islamic questions and answers, along with their topics, for 52 topic categories. We evaluate different state-of-the art deep learning models, both for word and sentence embeddings, comparing recurrent and transformer-based techniques, and performing extensive ablation studies to show the effect of each model choice. Our hierarchical model is based on pre-trained models, taking advantage of the recent advancement of transfer learning techniques, focused on Arabic language.



***Key words:***
*Islamic Fatwa, Natural Language Processing, Text Classification, Question Answering, Recurrent Neural Networks, Transformers.*


## 1. Introduction

Islam is characterized by a comprehensive set of immutable diving rules, representing the Islamic Law, or Sharia, which governs all aspects of Muslims lives. The Islamic jurisprudence or Fiqh represents the human interpretation of Sharia. Traditional theory of Islamic jurisprudence recognizes four sources of Sharia: the Quran, sunnah (authentic hadith), qiyas (analogical reasoning), and ijma (juridical consensus). Different legal schools, also called madhhabs, of which the most prominent are Hanafi, Maliki, Shafi school, Hanbali and Jafari. Classical jurisprudence was elaborated by private religious scholars, largely through legal opinions (fatwas) issued by qualified jurists (muftis). With the increase of Muslims population, representing 24.9% of the earth population in 51 countries, and the explosion of social media channels on one hand, and the scarcity of Muftis on the other hand, we have a supply demand problem, that calls for automation solutions, including Artificial Intelligence (AI). The potential of AI can be in automated Question-Answering (QA) systems, Chatbots and Question topic classification and routing.

A generic architecture of an automated QA system for Islamic Fatwas is shown in Fig.1. The system is based on two main stages: 1) Classify the question intent and 2) Generate the possible answer. Both stages need to be trained on a supervised dataset, that contains the Question, Intent and Answer. We built the largest dataset for Islamic Fatwas that contain 850,000 questions and answers, with 200,000 of them labelled by their topics. Details of this dataset will be covered in later sections. Intents can be as generic as a topic category, or specific as accurately determined question context. Modelling all intents of questions is not feasible for an open domain like Islamic Fatwas, hence we design a topic classifier that routes the question to an expert system specialized in certain areas of Fatwas. Such a system can be deployed in its early stages based on answers generated by Human Mufti's. This help to build the initial training set and might even be used on its own to relief the burden on Mufti's in real deployments, so that they focus on question in the areas of their domain expertise.


Manuscript received September 5, 2021
Manuscript revised September 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.9.37

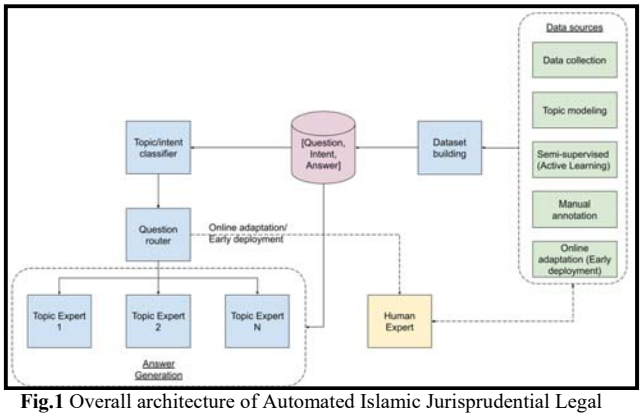


**Fig.1** Overall architecture of Automated Islamic Jurisprudential Legal Opinions Generation

In this work, we focus on the topic classifier component. We adopt the deep learning models in a hierarchical framework of three stages: 1) Pre-processing, 2) Features Extraction and 3) Decision and classification. The core of our system lies in the second stage, where we follow a hierarchical approach to obtain a question representation that can be used as features in the decision classifier. We first extract the words representations, followed by the sentence representation. Finally, the decision block is simply the classifier module that will generate a class label, in a multi-class problem setup. For each of the architecture blocks, we evaluate different options of the existing state-of-the art (SoTA) in natural language processing (NLP) models, where we evaluate different word embeddings and sentence embeddings models, focused on Arabic language, like Fasttext [1], AraVec [2] and AraBERT [3] and AraGPT-2 [4], making use of the recent advancement of transfer learning techniques in the NLP domain and using pre-trained models whenever possible.

We collect and build the largest dataset of Islamic Fatwas, from a diversity of the most popular Fatwa websites, official and non-official, spanning different geographical locations, accents, and backgrounds. The dataset includes 850,000 queries and answers, of which 200,000 entries are labelled for the topic class out of 52 possible topics categories. We use this data to train and test our models and evaluate different modelling options using ablation studies, to see the effect of every modelling choice. We use accuracy, precision, recall and F1 metrics in the classifier evaluation.

The rest of the paper is organized as follows: first we review the state-of- the art in NLP text classification, focused on Arabic language, then we present our methodology and models. Next, we present the experimental setup and results. And finally, we conclude with the discussion of the main outcomes and findings.

## 1.1 Background and related work

In the last few years, a lot of potential has been there for applied AI in Natural Language Processing (NLP). Following the Computer Vision (CV) field, NLP has reached the so-called "Image-Net moment" with the introduction of Transfer Learning and Transformers [5]–[7]. That potential is not fully unleashed in Low-Language Resources (LLR) like Arabic. Some attempts have been made such as [3], [8]–[12]. One important application of AI in NLP is the area of Personal Assistants, Chatbots and Question Answering (QA) systems, where AI delivers State-of-The-Art (SoTA) performance. Such applications are vital to domains where human experts can be overwhelmed by the high traffic of requests/questions, especially when the questions are repetitive, or at least could be clustered and routed to the proper expert ahead of time. Machine learning has been applied to categorisation of Quran and Hadith in [13], where basic text features are extracted, like TF-IDF, followed by traditional machine learning classifiers, like SVM and K-Nearest Neighbours. In this work, we follow a more sophisticated approach that leverage the power of hierarchical representations using deep learning methods, and the recent advancement of transfer learning, making of Arabic pre-trained models for both word and sentence embeddings.

**Transfer Learning in NLP.** One of the biggest challenges in natural language processing (NLP) is the shortage of training data. Because NLP is a diversified field with many distinct tasks, most task-specific datasets contain only a few thousand or a few hundred thousand human-labelled training examples. However, modern deep learning-based NLP models see benefits from much larger amounts of data, improving when trained on millions, or billions, of annotated training examples. To help close this gap in data, researchers have developed a variety of techniques for training general purpose language representation models using the enormous amount of unannotated text on the web (known as pre-training). The pre-trained model can then be fine-tuned on small-data NLP tasks like question answering and sentiment analysis, resulting in substantial accuracy improvements compared to training on these datasets from scratch.

In the field of computer vision, researchers have repeatedly shown the value of transfer learning—pre-training a neural network model on a known task, for instance ImageNet, and then performing fine-tuning—using the trained neural network as the basis of a new purpose-specific model. In recent years, researchers have been showing that a similar technique can be useful in many natural language tasks.

A basic form of transfer learning has been applied in NLP in the past few years, in the form of learning useful word representations; known as "Word Embeddings". Word Embeddings have seen advances recently being applied in FastText from FaceBook [1], and ELMo [14].

Pre-trained representations can either be context-free or contextual, and contextual representations can further be unidirectional or bidirectional. Context-free models such as word2vec or GloVe generate a single word embedding

representation for each word in the vocabulary. For example, the word "bank" would have the same context-free representation in "bank account" and "bank of the river." Contextual models, like BERT [5] and ELMo [14] instead generate a representation of each word that is based on the other words in the sentence. For example, in the sentence "I accessed the bank account," a unidirectional contextual model would represent "bank" based on "I accessed the" but not "account." However, BERT represents "bank" using both its previous and next context — "I accessed the ... account" — starting from the very bottom of a deep neural network, making it deeply bidirectional. ELMo learns contextual representations; the representation for each word depends on the entire context in which it is use. Moreover, it works at the character level, which reduces the Out-Of-Vocabulary (OOV).

Going beyond word representations, some new models appeared that focus on transfer learning on more useful architectures. Specifically, the model of encoder-decoder architecture started to take over in the field of Neural Machine Translation (NMT), like in seq2seq [15], which are based on BiLSTM models, and incorporate attention mechanisms, and the Transformer [6], which is fully based on attention gates, without any recurrent layers. Moreover, the learnt representations in that encoder, can be transferred to other tasks, like in ULMFiT [7], where a model is trained on large corpus for Neural Language Models (NLM), and then the backbone of the model is re-used to initialize a sentiment classification model on IMDB movie reviews. In BERT, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others.

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms; an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. BERT builds upon recent work in pre-training contextual representations — including Semi-supervised Sequence Learning, Generative Pre-Training, ELMo, and ULMFit. However, unlike these previous models, BERT is the first deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus.

A similar approach is used in Open AI GPT [16], which is a is a combination of two existing ideas: transformers and unsupervised pre-training. Open AI GPT works in two stages; first train a transformer model on a very large amount of data in an unsupervised manner — using language modelling as a training signal — then we fine-tune this model on much smaller supervised datasets to help it solve specific tasks.

**Potential in Arabic NLP.** Arabic language is considered among the Low-NLP Resources languages, unlike English. This calls for the need of both TL and MTL to help solving this issue. Looking on the literature today, there is a wide gap in applying the above techniques to Arabic NLP tasks. Transfer leaning of Word Embeddings was used in AROMA [11], using learnt embeddings from QALB dataset, to perform sentiment classification task. There is a high potential in applying the SOTA discussed above in the tasks of Arabic Opinion Mining (OMA) and Emotion Recognition. More recently, different pre-trained models for Arabic are released, like AraBERT [3] and AraGPT-2 [4].

## 2. Methodology

We follow a hierarchical approach shown in Fig.2. First, we perform text pre-processing to normalize, clean and vectorize the text. Then we perform features extraction in a hierarchical way; where we first extract word representations, followed by sentence representation. Finally, the classifier acts on the sentence representation as features vector to produce the final topic label. The whole problem is formulated as a multi-class classification problem.

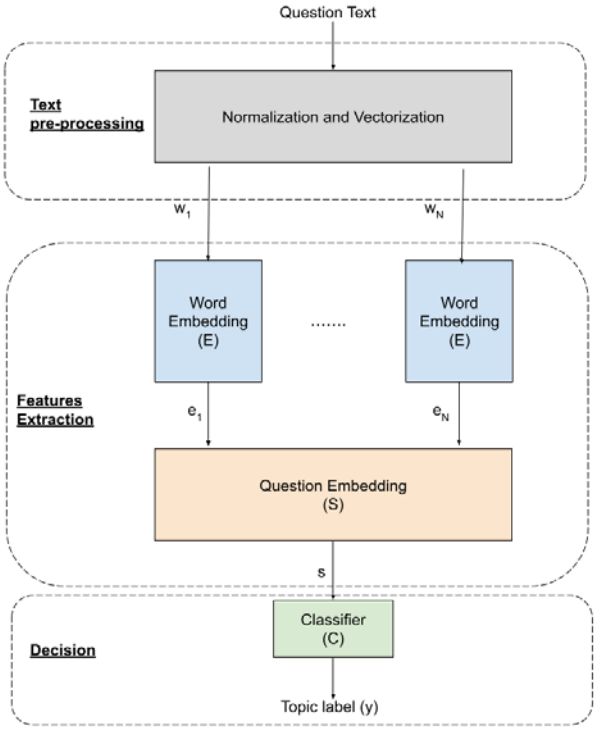


**Fig.2** Hierarchical Deep Learning Text Classification Model

### 2.1 Text pre-processing

The aim of this first stage is to clean and vectorize the text into numerical indices. The first step is to clean and

normalize the text. An important factor in this process is to reduce the variability and noise in the text, such that, only the important tokens are kept. The following pipeline was applied: 1) Special and non-Arabic characters removal. 2) Arabic Diacritics removal. 3) Punctuation removal. 4) Numbers removal. 5) Stop words removal (using NLTK Arabic set). 6) Stemming using ISRIStemmer for Arabic.
Based on the resulting corpus of text, a vocabulary table $V$ can be built, out of the unique tokens that will result. The more efficient the cleaning process, the smaller the vocabulary size is. A large vocabulary size will affect the model choice and size later, and hence we want to keep as small and efficient as possible. On the other hand, a small vocabulary size, might result in many Out-Of-Vocabulary (OOV) indices in the vectorization process. The next step is to vectorize the cleaned text into numerical tokens indices, $w_i \in 1,2,\dots V$ is the index of the word, selected from a vocabulary range $|V|$. The length of the sequence of tokens is padded with zeros to a maximum of $N$ tokens.

## 2.2 Word Embedding

Word Embedding Look-up Table (LUT) $E \in R^{V \times d}$, where $d$ is the embedding dimension and $V$ is the vocabulary size. The entries of this table are the words representations to be learnt, and thus they represent the learnable parameters of this block. Further, they can be pre-trained and fine-tuned as will be described in the next sections. The result of the loop up operation is an embedding vector $e_i \in R^d$. For mathematical convenience, the look-up operation is usually done as a dot product operation, which enables an end-to-end graph that can be trained using gradient descent. In this case the word indices are converted into One-Hot-Encoded (OHE) vectors, $\hat{e} \in R^V$, which is sparse vector that has all zeros, except at the index of the of $e_i$. Now the embedding vector can be obtained as a simple dot product $e_i = \hat{e} \odot E$.

## 2.3 Transfer learning and Pre-trained Embeddings

The word embeddings block is parametrized by the word vectors $E \in R^{V \times d}$, which are initialized randomly, and fine-tuned as part of the model optimization using gradient descent methods. It is also possible to use pre-trained embeddings tables, and fine-tune them, instead of random initializations. For that, we used two options of pre-trained embeddings: 1) Aravec [2] and 2) Fasttext [1].

## 2.4 Question Embedding.

The sentence embedding merger block $S$ aggregates all the word embeddings vectors into one representation $s \in R^{d_s}$, where $d_s$ is the sentence embedding vector dimension. The sentence merge is a neural network, that is parametrized by set of weights to be learnt according to the final loss. In the next sections, we will show the different sentence embeddings models we used.

## 2.5 Models

**Bag-of-Words.** All bag of words models had the same model architecture and had a vocabulary size of 2000. The architecture consisted of 2 Dense layers with 1000 nodes followed by 512 node layers and then the classification layer. Dropouts were applied after every Dense Layer to reduce overfitting. The bag of words models used the following text features: binary, count, frequency and TFIDF. Moreover, we also evaluate a BoW vectors, where we use an Embedding layer for each input word. This requires padding the input sentence to a maximum of 250 words. The vocabulary size is also 2000 words and the embedding shape is 300. Embeddings layers for BoW vectors model are trained from scratch; i.e. initialized with random weights sampled from a normal distribution.

**Recurrent based models.** The separate words vectors $e_i \in R^d$ can be aggregated using a recurrent neural network (RNN), which acts as a state machine that sequentially processes the token vector, resulting in a hidden state that is updated with each token in the sequence. The final hidden state can then be used to represent the whole sequence. A single layer RNN is parametrized by input weights $U$, hidden transition weights $V$ and output weights $O$ as shown in Fig.3. It is possible to stack more of such layers in a straightforward way using the hidden states representations. We used LSTM [17] and GRU [18] recurrent models.

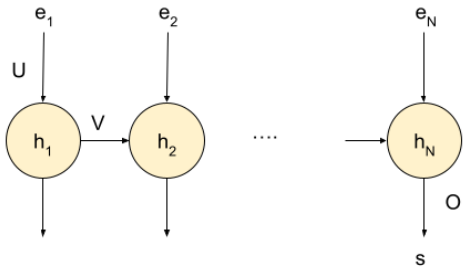


**Fig.3** Single layer Recurrent Neural Network

This model was implemented in three different flavors, we used an embedding layer to embed the words from scratch and we used two pre-trained Arabic word embedding models AraVec and Fasttext, with 300 embedding dimension. We used single layer LSTM, and then layers were added. The first one is another LSTM layer and the second one is a dense layer that was put between the LSTM and the output layer, all resulting in 100 dimensions sentence embedding. The GRU model is the same network as the LSTM network except that the LSTM was replaced

with a GRU layer, with 1 and 2 layers. It also had 100 dimensions.

**Transformer based models.** We use attention-based sentence embedding models. We start with BERT [5]. BERT is based on the full attention mechanism introduced in [6] for sequence-to-sequence models, where we have and encoder-decoder architecture. The encoder is based on full self-attention mechanism, resulting in an embedding vector for each input token of the input sentence. However, for sentence representation, we need only one vector for the whole sentence. To work this out, BERT introduces a new CLS token at the input, that has learnable embeddings. The output tokens vectors can then be ignored, except the first one, which corresponds to the CLS token, and hence holds a representation for the whole sentence as shown in Fig.4. The whole encoder is pre-trained on different up-stream tasks, like Next Sentence Prediction (NSP) and Masked Language Modeling (MLM), and can then be fine tuned to any downstream task, like sentence classification.

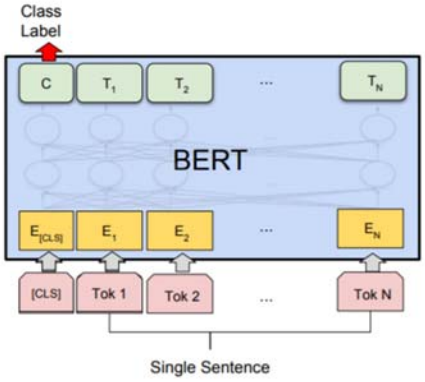


**Fig.4** BERT for single sentence classification [5]

We used several pre-trained transformer models. We start with AraBERT [3] is a BERT transformer that was trained on Arabic data with a tweaked tokenizer that is specific for the Arabic words and Arabic word compounds. We used a hugging face classification library to load AraBERT-base model and train on the dataset. Hugging face pools all output embeddings and adds one dense layer and one classification layer. We also used AraBERT's pre-process function that cleans the text and put it in the structure that AraBERT's tokenizer can read. We used the same process as we used in AraBERT but with the AraGPT-2 model [4]. It did not have a padding token, so we had to use the EOF token as the padding token in this training trial. CLS Token is considered a token that has the document embedding of the input. We used the CLS as an embedding and added one dense layer and one classification layer. Here the transformer is frozen.

Finally, we tried Bidirectional LSTM with frozen transformer; where we used all embeddings of the transformer and fed it into a bidirectional LSTM, keeping the transformer weights frozen.

### 2.6 Decision classifier

The classifier model $C$ can now act on the sentence embedding $s$ as the features vector and produce the class label $y \in 1,2,\dots,K$, where $K$ is the number of possible classes. The classifier also is parameterized by set of learnable weights to be optimized according to the final loss. Given the sentence embedding features vector $s \in R^{d_s}$, we use a fully connected layer $w \in R^{d_s \times K}$, resulting in class scores $s_i = \mathrm{s} \odot w$, where $s_i \in 1,2,\dots,K$. the final class label $y_i \in 1,2,\dots,K$ can be obtained using a softmax operation as follows:

$$y_i = \frac{e^{s_i}}{\sum_{i=1}^{K} e^{s_i}}$$

**Loss and Optimization.** The hierarchical model after the text normalization and pre-processing step represents an end-to-end graph, that starts with the word's indices, $w_i \in 1,2,\dots V$ of maximum $N$ tokens, through learnable word embeddings table $E \in R^{V \times d}$, which is modelled as fully connected neural network. This is then followed by the parametrized sentence embedding neural model $S$, feeding the classier $C$ resulting in the final question labels $y \in 1,2,\dots,K$. We can now write the loss according to the categorical cross entropy loss minimization for our multi-class problem. We can formulate the log likelihood loss as follows, given the ground truth label of the question as $t \in 1,2,\dots,K$

$$\mathcal{L}(E,S,C) = -\sum_{i=1}^{K} t_i log y_i$$

The whole model is a differentiable neural network model, and hence the loss is a function of the hierarchical model parameters $\mathcal{L}(E,S,C)$. To find the different models' parameters, we can optimize the loss end-to-end using Stochastic Gradient Descent optimization:

$$\mathcal{L}^* = min\ \mathcal{L}(E,S,C)$$

$$E^*,S^*,C^* = argmin_{E,S,C}\mathcal{L}$$

## 3. Experiments and Results

### 3.1 Experimental Setup

For the classifier models, we evaluate three families: The Bag-of-Words (BoW) and Sequence models: recurrent based or transformer-based models. All bag of words models had the same model architecture and had a vocabulary size of 2000. The architecture consisted of 2 Dense layers with 1000 nodes followed by 512 node layers

and then the classification layer. Dropouts were applied after every Dense Layer to reduce overfitting. The bag of words models used the following text features: binary, count, frequency and TFIDF.

Moreover, we also evaluate a BoW vectors, where we use an Embedding layer for each input word. This requires padding the input sentence to a maximum of 250 words. The vocabulary size is also 2000 words, and the embedding shape is 300. Embeddings layers for BoW vectors model are trained from scratch, i.e., initialized with random weights sampled from a normal distribution.

We evaluate both LSTM and GRU models, with 1 and 2 layers. For Transformer Networks Topics Classifier, we used a hugging face classification library to load AraBERT-base model and train on the dataset. We also used AraBERT's pre-process function that cleans the text and put it in the structure that AraBERT's tokenizer can read. This model was implemented in three different flavors, we used an embedding layer to embed the words from scratch and we used two pre-trained Arabic word embedding models AraVec (Soliman, Eissa, & El-Beltagy, 2017) and Fasttext (Athiwaratkun, Wilson, & Anandkumar, 2018). All the three implementations used the same architecture, but the pre-trained embedding had 300 dimensions instead of 300 and the embedding was non-trainable.

## 3.2 Dataset

The details of the dataset collection are shown in Table 1, of around 850,000 Fatwas (questions and answers). We crawl the popular websites of Islamic Fatwa, being official, like Al-Ifta-SA, Dar-al-ifta-EG and Al-ifta-JO, or non-official like islamway, islamweb,..etc. Those websites span different countries and geographical locations, accents, and backgrounds. We crawl for question, answer, topic and date. For Arabic AskFM, we extend the one in (AskFM98k, n.d.) to include 604,000 fatwas, by crawling the full website. A special type of QA is found in islamonline, where we treat the articles titles as questions, and the bodies as answers, since they form the basic and frequently asked questions in Islamic Fatwa.

**Table 1** Dataset information, statistics, and sources

| ***Dataset*** | ***Question/Answers*** | ***Topics*** | ***Dates*** |
|---|---|---|---|
| Al-ifta-SA (AiIftaSA, n.d.) and Dar-al-ifta-EG (DarAlIftaEG, n.d.) | 3,450 | Yes | Yes |
| AskFM (AskFM98k, n.d.) | 604,184 | N/A | N/A |
| Islamweb (Islamweb, n.d.) | 126,000 | Yes | Yes |
| Islamway (Islamway, n.d.) | 15,060 | N/A | Yes |
| Islamonline (Islamonline, n.d.) | 3,100 | Yes | N/A |
| binbaz (Binbaz, n.d.) | 28,226 | Yes | N/A |
| binothaimeen (Binothaimeen, n.d.) | 2,157 | Yes | N/A |
| AlFawzan (Alfawzan, n.d.) | 2,000 | N/A | Yes |
| Islamqa (Islamqa, n.d.) | 30,780 | Yes | Yes |
| Fatwapedia (Fatwapedia, n.d.) | 34,661 | Yes | N/A |

The topics and dates are not applicable or present for some websites. For topics categories, we have 200,000 questions, labelled with the corresponding topics. Topics can be in the form of nested topics classes. We aggregate over all the available topics.

## 3.3 Metrics

We formulate our problem as multi-class classification problem, for $K = 52$ categories. For that, we evaluate our system based on two metrics.

**Average Accuracy.** The accuracy per class is defined as follows:

$$Accuracy = \frac{TP + TN}{P + N}$$

Where TP=True Positives, TN=True Negatives, P = Total positive samples and N=Total negative samples. For binary classification case, True and False samples are easily defined. For multi-class, True samples are the ones belonging to the class being score, and False are any samples from other classes. Same goes for Positives and Negatives. This results in an accuracy score per class of the $K = 52$ categories. To evaluate a model, we average the accuracies over the 52 classes:

$$AvgAcc = \sum_{1=1}^{i=K} Accuracy_i$$

**Macro-Average F1-score.** The F1-score per class is defined as follows:

$$F_1 = 2 \times \frac{PR \times RE}{PR + RE}$$

Where PR=Precision and RE=Recall, defined as:

$$PR = \frac{TP}{TP + FP}$$

$$RE = \frac{TP}{TP + FN}$$

This results in an F1-score per class of the $K = 52$ categories. To evaluate a model, we average the class F1-scores over the 52 classes:

$$MacroF1 = \sum_{i=1}^{i=K} F_1^i$$

### 3.4 Results

The collective results are shown in **오류! 참조 원본을 찾을 수 없습니다.**. Three models' families are evaluated according to the experiments design: BoW, recurrent and transformer-based models. Whenever applicable, different word Embeddings options are listed. At a high level, the results suggests a clear advantage of the transformer-based models. Also, transfer learning shows a consistent advantage, both on the word embedding level, as shown for AraVec and Fasttext results, and on the sentence embedding level for the cases of AraBERT and AraGPT-2. Detailed discussion and analysis of the results are tackled in the next section.

Table 2 Topics Classifiers Baseline Results

| ***Model*** | ***Accuracy (%)*** | ***Precision (%)*** | ***Recall (%)*** | ***$F_1$ (%)*** |
|---|---|---|---|---|
| *BoW-Binary* | *53.3* | *47* | *41* | *42* |
| *BoW-Count* | *53.4* | *48* | *39* | *41* |
| *BoW-Frequency* | *51* | *39* | *30* | *30* |
| *BoW-TF-IDF* | ***53.5*** | ***44*** | ***42*** | ***43*** |
| *BoW Vectors* | *47* | *36* | *34* | *34* |
| *1-Layer LSTM* | *54* | *44* | *38* | *39* |
| *1-Layer GRU* | ***55*** | ***45*** | ***40*** | ***42*** |
| *2-Layer LSTM* | *53* | *46* | *36* | *38* |
| *2-Layer GRU* | *54* | *45* | *38* | *43* |
| *AraBERT* | ***70*** | ***59*** | ***57*** | ***56*** |

**Table 3** Comparison between the effect of different word Embeddings models for the different topic classifiers

| *Model* | *Embedding* | | | |
|---|---|---|---|---|
| | ***None*** | ***From scratch*** | ***AraVec*** | ***Fasttext*** |
| ***Bag-of-Words Models*** | | | | |
| ***BoW-Binary*** | *53.3%* | - | - | - |
| ***BoW-Count*** | *53.4%* | - | - | - |
| ***BoW-Frequency*** | *51%* | - | - | - |
| ***BoW-TF-IDF*** | ***53.5%*** | - | - | - |
| ***BoW Vectors*** | - | *47%* | - | - |
| ***Recurrent-based Models*** | | | | |
| ***1-Layer LSTM*** | - | *52%* | *44%* | *58%* |
| ***1-Layer GRU*** | - | *53%* | *49%* | ***62%*** |
| ***2-Layer LSTM*** | - | *50%* | *54%* | *55%* |
| ***2-Layer GRU*** | - | *56%* | *52%* | ***62%*** |
| ***Transformer-based Models*** | | | | |
| ***AraBERT*** | ***70%*** | - | - | - |
| ***AraGPT-2*** | *64%* | - | - | - |

## 4. Discussion

**Effect of sentence embedding.** Bag-of-words models: On the other hand, BoW lacks the advantage of context, hence sequence models, like recurrent and transformers models, outperform them, by 2-8%, excluding the effect of pre-trained word vectors. However, for text classification, keywords of vocabulary could be more critical. Hence, we do not see huge gap in performance.

**Recurrent models.** For recurrent based models, GRU layers are outperforming LSTM layers by 4-7%. Adding

extra layers for both options does not seem to have an advantage.

**Transformer-based models.** Transformer based models were introduced as a replacement to recurrent models. In BERT, the idea of pre-trained language models was coupled with the full attention bi-directional transformers. The dependence on a pre-trained language model raises the question of the efficiency of off-the-shelf models for multi-lingual text classification. Thus, language specific, pre-trained models were introduced in AraBERT and AraGPT-2. The results show a clear advantage of Arabic specific language models, with 20-26% advantage over the generic model, fine-tuned on our task. AraBERT is outperforming AraGPT-2 by 6%.

**Effect of Transfer learning pre-trained embeddings.** Transfer learning in NLP can be viewed at the word level, with pre-trained word Embeddings like fasttext, or language models, like in BERT and GPT. Thus, when it comes to pre-trained embeddings with recurrent models, or pre-trained language models with transformer, we see a higher 9-16% advantage over the BoW models. The effect of pre-trained language models in AraBERT is more dominant, giving the top score of 70%.

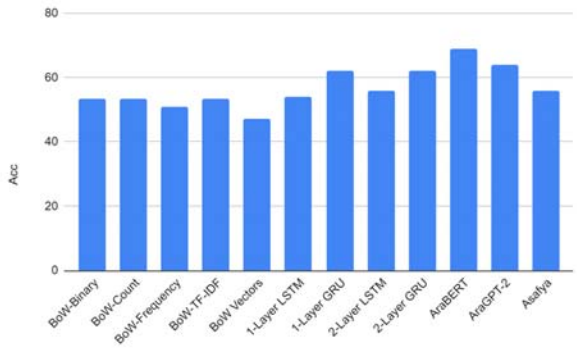


**Fig.5** Sentence embeddings models comparison

**Effect of text features.** While the differences are small, we can see an advantage for the frequency and TF-IDF features, probably because of the normalization effect they introduce to the features. Following is the binary features model, which also keeps a 0/1 hard-normalized features. Finally, the least performer is the count model, which does not perform any normalization, giving false advantage to the frequent words. The count model is not far from the other, since we perform rigorous text cleaning, removing irrelevant words. The BoW vectors model is least performer because it starts from randomly initialized Embeddings.

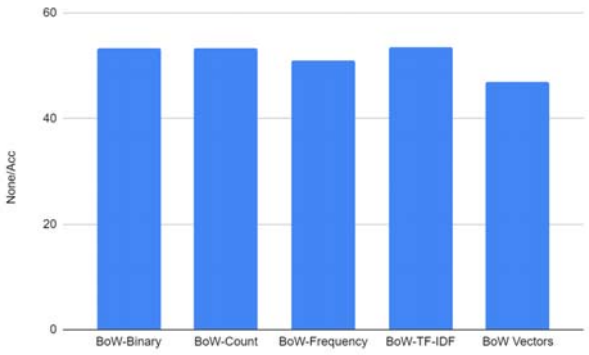


**Fig.6** Effect of text features

**Effect of pre-trained word embeddings.** The effect of pre-trained Embeddings s studied with the recurrent based models; GRU and LSTM. AraVec performed poorly because it has a lot of Out-of-Vocabulary (OOV) words. Fasttext was the top performer. The way fasttext works is by treating each word as a bag of character n-grams (from 3 to 6 in practice). Each word vector is represented by summing the vectors of its character n-grams plus a specific word vector for the word itself.

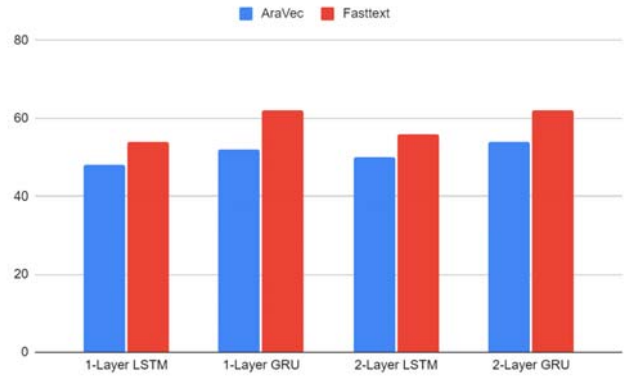


**Fig.7** Effect of pre-trained word embeddings

**Effect of Classifier Choice (SVM vs. Softmax).** In order to evaluate the effect of the classifier decision layer, we evaluate the softmax with categorical cross entropy classifier, versus an SVM classifier, that uses One-Versus-Rest (OVR), heuristic for multi-class classification. In this experiment, we use the embeddings from AraBERT model, and apply the different decision layers. As shown in Fig.8, the results of a softmax classifier outperform the SVM classifier by around 7% in accuracy and 6% in F1 score. The reason is that SVM is a binary classifier by nature and uses heuristics like OVR to transform it into multi-class setups. In our case, we have a a multi-class problem, with large number of classes (52 categories). On the other hand, cross entropy loss with negative log likelihood minimization

works naturally for multi-class problems, where a true probability distribution is obtained over the class labels.

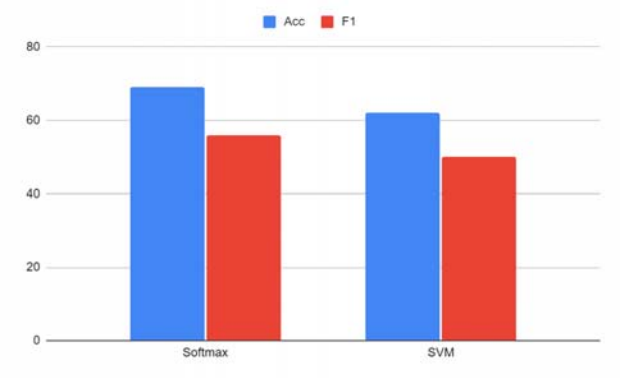


**Fig.8** SVM vs. Softmax

## 5. Conclusion

In this work, we presented a hierarchical deep learning text classifier for automatic categorisation of Islamic Fatwa questions. Our model is based on hierarchical representations of words, followed by sentence embeddings, and finally a decision layer for classification of the question topic. We extensively evaluated our model for different modelling choices, evaluating two pre-trained word embeddings, three sentence embeddings models and two decision layers classifiers. We collect and release the largest publicly available Islamic Fatwa dataset, annotated for questions, answers and topics. Our study shows clear advantage of pre-trained models and transfer learning, both on word and sentence levels. Also, Arabic specific pre-trained models have the best performance. Finally, we show the advantage of state-of-the art transformer-based models for our application of Islamic Fatwa categorisation reaching 70% accuracy and 56% F1 score performance.

### Acknowledgments

The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding their research work through the project number 20-UQU-IF-P3-001.

## References

[1] B. Athiwaratkun, A. G. Wilson, and A. Anandkumar, "Probabilistic fasttext for multi-sense word embeddings," *arXiv Prepr. arXiv1806.02901*, 2018.

[2] A. B. Soliman, K. Eissa, and S. R. El-Beltagy, "Aravec: A set of arabic word embedding models for use in arabic nlp," *Procedia Comput. Sci.*, vol. 117, pp. 256–265, 2017.

[3] W. Antoun, F. Baly, and H. Hajj, "Arabert: Transformer-based model for arabic language understanding," *arXiv Prepr. arXiv2003.00104*, 2020.

[4] W. Antoun, F. Baly, and H. Hajj, "AraGPT2: Pre-Trained Transformer for Arabic Language Generation," Dec. 2020, Accessed: Jul. 05, 2021. [Online]. Available: http://arxiv.org/abs/2012.15520.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv Prepr. arXiv1810.04805*, 2018.

[6] A. Vaswani, "Attention Is All You Need," no. Nips, 2017.

[7] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv Prepr. arXiv1801.06146*, 2018.

[8] M. Djandji, F. Baly, H. Hajj, and others, "Multi-Task Learning using AraBert for Offensive Language Detection," in *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, 2020, pp. 97–101.

[9] A. M. Abu Nada, E. Alajrami, A. A. Al-Saqqa, and S. S. Abu-Naser, "Arabic Text Summarization Using AraBERT Model Using Extractive Text Summarization Approach," 2020.

[10] A. Al Sallab, M. Rashwan, H. Raafat, and A. Rafea, "Automatic Arabic diacritics restoration based on deep nets," in *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, 2014, pp. 65–72.

[11] A. Al-sallab, R. Baly, H. Hajj, K. B. Shaban, W. El-hajj, and G. Badaro, "AROMA : A Recursive Deep Learning Model for Opinion Mining in Arabic as a Low Resource Language," vol. 16, no. 4, 2017.

[12] A. Magooda *et al.*, "RDI-Team at SemEval-2016 task 3: RDI unsupervised framework for text ranking," 2016.

[13] N. A. P. Rostam and N. H. A. H. Malim, "Text categorisation in Quran and Hadith: Overcoming the interrelation challenges using machine learning and term weighting," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 33, no. 6, pp. 658–667, Jul. 2019, doi: 10.1016/j.jksuci.2019.03.007.

[14] M. E. Peters *et al.*, "Deep contextualized word representations," *arXiv Prepr. arXiv1802.05365*, 2018.

[15] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv Prepr. arXiv1409.0473*, 2014.

[16] T. B. Brown *et al.*, "Language models are few-shot learners," *arXiv Prepr. arXiv2005.14165*, 2020.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp.

1735–1780, 1997.

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," *arXiv Prepr. arXiv1502.02367*, 2015.

[1] B. Athiwaratkun, A. G. Wilson, and A. Anandkumar, "Probabilistic fasttext for multi-sense word embeddings," *arXiv Prepr. arXiv1806.02901*, 2018.

[2] A. B. Soliman, K. Eissa, and S. R. El-Beltagy, "Aravec: A set of arabic word embedding models for use in arabic nlp," *Procedia Comput. Sci.*, vol. 117, pp. 256–265, 2017.

[3] W. Antoun, F. Baly, and H. Hajj, "Arabert: Transformer-based model for arabic language understanding," *arXiv Prepr. arXiv2003.00104*, 2020.

[4] W. Antoun, F. Baly, and H. Hajj, "AraGPT2: Pre-Trained Transformer for Arabic Language Generation," Dec. 2020, Accessed: Jul. 05, 2021. [Online]. Available: http://arxiv.org/abs/2012.15520.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv Prepr. arXiv1810.04805*, 2018.

[6] A. Vaswani, "Attention Is All You Need," no. Nips, 2017.

[7] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv Prepr. arXiv1801.06146*, 2018.

[8] M. Djandji, F. Baly, H. Hajj, and others, "Multi-Task Learning using AraBert for Offensive Language Detection," in *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, 2020, pp. 97–101.

[9] A. M. Abu Nada, E. Alajrami, A. A. Al-Saqqa, and S. S. Abu-Naser, "Arabic Text Summarization Using AraBERT Model Using Extractive Text Summarization Approach," 2020.

[10] A. Al Sallab, M. Rashwan, H. Raafat, and A. Rafea, "Automatic Arabic diacritics restoration based on deep nets," in *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, 2014, pp. 65–72.

[11] A. Al-sallab, R. Baly, H. Hajj, K. B. Shaban, W. El-hajj, and G. Badaro, "AROMA : A Recursive Deep Learning Model for Opinion Mining in Arabic as a Low Resource Language," vol. 16, no. 4, 2017.

[12] A. Magooda *et al.*, "RDI-Team at SemEval-2016 task 3: RDI unsupervised framework for text ranking," 2016.

[13] N. A. P. Rostam and N. H. A. H. Malim, "Text categorisation in Quran and Hadith: Overcoming the interrelation challenges using machine learning and term weighting," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 33, no. 6, pp. 658–667, Jul. 2019, doi: 10.1016/j.jksuci.2019.03.007.

[14] M. E. Peters *et al.*, "Deep contextualized word representations," *arXiv Prepr. arXiv1802.05365*, 2018.

[15] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv Prepr. arXiv1409.0473*, 2014.

[16] T. B. Brown *et al.*, "Language models are few-shot learners," *arXiv Prepr. arXiv2005.14165*, 2020.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," *arXiv Prepr. arXiv1502.02367*, 2015.

**Wesam H alsabban** received the B.Sc. degree in electrical and computer engineering from Umm Al-Qura University, Makkah, Saudi Arabia, in 2000, and the M.Sc. and PhD degrees in engineering systems from Queensland University of Technology , Brisbane, Australia, in 2009 and 2014, respectively. Currently Dr. Wesam is an Assistant Professor in the Computer Engineering department at Umm Al-Qura University, Makkah, Saudi Arabia. His research interests include artificial intelligent, robotic, and deep learning.

**Saud S. Alotaibi** is an associate professor of Computer Science at the Umm Al-Qura University, Makkah, Saudi Arabia. He received the Bachelor of Computer Science degree from King Abdul Aziz University, Jeddah, KSA, in 2000, the Master's degree in Computer Science from King Fahd University, Dhahran, KSA, in May 2008, the Ph.D. degrees in Computer Science from Colorado State University, Fort Collins, USA, in August 2015. From January 2009 to 2017, he worked as a Vice Deputy of IT Center at Umm Al-Qura University, Makkah, KSA. Then, he was designated as Vice COE of the Smart Campus. Currently, he is the dean of College of Computer and Information Systems at the same university. His current research interests include Emotional Intelligence, Data Mining, Natural Language Processing, Machine Learning, Deep Learning, Computer Networks, Wireless Sensor Networks and Network Security.

**Abdullah Tarek Farag** received B.Sc. degree in computer science from MSA university in 2018. He then started working as a computer vision engineer for a year and a half. Then worked as a data scientist and an NLP engineer.

**Omar Essam Rakha** received the B.Sc. degree from the faculty of engineering, Ain shams university and has since been working as an AI research engineer with specific interest in NLP.

**Ahmad A. Al Sallab** received his B.Sc. from the Faculty of Engineering, Cairo University in Electronics and Communications. He acquired his M.Sc. and Ph.D. in 2009 and 2013 from Cairo University in Artificial intelligence. Ahmad has 17 years of practical experience, where he worked for reputable multi-national organizations in the industry like Intel and Valeo. He has over 40 publications and book chapters in top IEEE and ACM journals and conferences, in addition to 8 patents filed in European and German patent offices, with applications in Speech, NLP, Computer Vision and Robotics.

**Majid Alotaibi** received Ph.D. from The University of Queensland, Brisbane, Australia, in 2011. Currently, he is an associate professor with the Department of Computer Engineering, Umm AlQura University, Makkah, and the Co-founder of the SMarT Lab. His current research interests include mobile computing, mobile and sensor networks, wireless technologies, IoT in Healthcare, Smart Cities, ad-hoc networks, computer networks (wired/wireless), RFID, antennas and propagation, and radar. Digital transformations: Frameworks, Strategy, Enterprise Architecture, Governance, Technologies, and Data.