Information Sciences 615 (2022) 90–102

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Building siamese attention-augmented recurrent convolutional neural networks for document similarity scoring

Sifei Han [a], Lingyun Shi [a], Russell Richie [a], Fuchiang R. Tsui ("Rich") [a,b,*]

[a] Tsui Laboratory, Department of Biomedical and Health Informatics, Children's Hospital of Philadelphia, 2716 South Street, Philadelphia, Pennsylvania, USA
[b] Perelman School of Medicine, University of Pennsylvania, 3400 Spruce Street, Suite 680 Dulles, Philadelphia, Pennsylvania, USA

A R T I C L E   I N F O

Article history:
Received 13 April 2022
Received in revised form 8 July 2022
Accepted 3 October 2022
Available online 7 October 2022

Keywords:
Attention neural network
Deep learning
Machine learning
Natural language processing
Information retrieval
Text similarity

A B S T R A C T

Automatically measuring document similarity is imperative in natural language processing, with applications ranging from recommendation to duplicate document detection. State-of-the-art approach in document similarity commonly involves deep neural networks, yet there is little study on how different architectures may be combined. Thus, we introduce the Siamese Attention-augmented Recurrent Convolutional Neural Network (S-ARCNN) that combines multiple neural network architectures. In each subnetwork of S-ARCNN, a document passes through a bidirectional Long Short-Term Memory (bi-LSTM) layer, which sends representations to local and global document modules. A local document module uses convolution, pooling, and attention layers, whereas a global document module uses last states of the bi-LSTM. Both local and global features are concatenated to form a single document representation. Using the Quora Question Pairs dataset, we evaluated S-ARCNN, Siamese convolutional neural networks (S-CNNs), Siamese LSTM, and two BERT models. While S-CNNs (82.02% F1) outperformed S-ARCNN (79.83% F1) overall, S-ARCNN slightly outperformed S-CNN on duplicate question pairs with more than 50 words (39.96% vs. 39.42% accuracy). With the potential advantage of S-ARCNN for processing longer documents, S-ARCNN may help researchers identify collaborators with similar research interests, help editors find potential reviewers, or match resumes with job descriptions.

1. Introduction

In various applications, knowing the similarity between documents is essential to users' needs. An editor might want to know if a paper submission is similar enough to others to warrant concerns of plagiarism, an avid book reader might want to find books similar to one they enjoyed, and a user on a help website might want to know if questions similar to their own have already been asked and answered. Crowdsourced annotation of the similarity between documents can be useful in some cases, but still requires enormous collective human effort, and is inapplicable in some cases where privacy is paramount.

Accordingly, the last few decades have seen intense development of natural language processing (NLP) methods to automatically score document pairs on their similarity. **State-of-the-art approach on various document similarity bench-**

* Corresponding author at: Tsui Laboratory, Department of Biomedical and Health Informatics, Children's Hospital of Philadelphia, 2716 South Street, Philadelphia, Pennsylvania, USA.
E-mail address: tsuif@chop.edu (F.R. Tsui).

https://doi.org/10.1016/j.ins.2022.10.032
0020-0255/© 2022 Elsevier Inc. All rights reserved.

marks is commonly achieved by deep neural networks (DNNs), and in particular by convolutional neural networks (CNNs), long short-term memory (LSTM) networks, feedforward attention networks (*i.e., transformers like Bidirectional Encoder Representations from Transformers [BERT]), and/or Siamese neural networks (S-NNs)*. For example, the original BERT model [1] was tested on the Quora Question Pairs (QQP) dataset, achieving 71.2% F1 with BERT_base and 72.1% with BERT_large. Yaghoobzadeh et al. [2] used BERT to embed a document, and fed these embeddings into a k-NN (k-Nearest Neighbor) model, and achieved an AUC of 80% in detecting duplicate questions in the QQP dataset. McCreery et al. [3] fine-tuned XLNet to predict medical question–answer pairs, and then transferred this model to the task of detecting duplicate Aedical questions, where it achieved 82.6% accuracy. Imtiaz et al. [4] used Siamese LSTM's with Manhattan distance for measuring similarity, and achieved 91.14% accuracy on QQP. Zheng et al. [5] used a CNN model to detect the similarity of medical texts (to enhance downstream information retrieval), achieving a macro F1-score of 93.5% on their in–house dataset of clinical reports.

While each of the popular network components or architectures – CNN, LSTM, attention, Siamese – has shown value for document representation and similarity when used in isolation, **it is less well-known how powerful the combination of all of these network components might be**. We believe this is a crucial knowledge gap, since different components are thought to serve different purposes. That is, convolution allows extraction of local features while the LSTM allows extraction of global features [6]. Thus, having both local and global features may be important for developing a system that can measure similarity of both short and long texts. Attention, on the other hand, allows token representations to be influenced by the most relevant other tokens in a sentence, while the Siamese architecture allows (among other things) efficient inference and robustness to class imbalance.

Despite the potential power of combining these components, to our knowledge, there is only one previous study that combines convolution, recurrence (LSTM), and attention in Siamese neural networks for document similarity: Huang et al. (2020) [7]. In one subnetwork (dubbed ARC-2) of Huang et al.'s Siamese architecture, convolutional and LSTM layers are applied in parallel and merged, then passed through a fully connected layer and then an attention layer. This is, of course, only one of many possible ways that these components could be combined, and so it is essential to test different possible combinations of these components, since it is largely an empirical question which combinations perform better than others.

In this paper, **we therefore propose and test a new architecture for measuring document similarity – the Siamese attention-augmented recurrent convolutional neural network (S-ARCNN)**. S-ARCNN extracts both local and global representations from a document. It first applies a bi-LSTM, and then sends extracted representations to two separate modules: a module with convolution to extract local features, and another module that extracts global features by max and average pooling, attention weighting, and concatenating the last states of the forward and backward LSTM's. We then concatenate these local and global features to form a single sentence representation. We compare S-ARCNN to two standard (DNN) models, Siamese Long Short-Term Memory (S-LSTM) and Siamese Convolutional Neural Networks (S-CNN), on a standard benchmark dataset of sentence similarity, the Quora Question Pairs dataset of duplicate questions. We hypothesize that our proposed model S-ARCNN can outperform current state-of-the-art DNNs in measuring document similarity. Overall, then, our contributions are:

- A novel deep neural network architecture combining recurrence, convolution, and attention in Siamese neural networks for measuring document similarity
- Comparisons of this architecture on the Quora Question Pairs dataset of duplicate questions, to more traditional architectures including Siamease LSTM, Siamese CNN, and a range of state-of-the-art transformer architectures.
- Evaluation of our and competitor models on longer question pairs ($\geqslant$ 50 words), where our model out-performs others on duplicate questions.

The rest of the paper is organized as follows. Section 2 briefly describes background on measuring text similarity. Section 3 and Section 4 present information about the dataset and models we test, respectively. The experiment settings and the evaluation on the Quora Question Pairs (QQP) dataset are described in Section 5. Section 6 contains discussion. Finally, we give our conclusion and some suggestions for future work in Section 7.

2. Background on measuring text similarity

Measuring document similarity is one of the oldest tasks in natural language processing. The oldest, most conventional approaches generally rely simply on representing documents with vectors of word frequency, known as the Bag of Words (BoW). For example, the two sentences/documents *I like cats* and *I love cats and dogs* would be represented as vectors [1, 1, 1, 0, 0, 0] and [1, 0, 1, 1, 1, 1], respectively, indicating their count of the six word types in the two sentences: *I, like, cats, love, and,* and *dogs*. Similarity between such vectors can then be computed in various ways, e.g., cosine similarity [8], Minkowski distance [9], and Jaccard similarity [10]. Additional pre-processing steps can be used to improve performance, like lemmatization (mapping both *cats* and *cat* to *cat*), removing low information stop words like *the, which*, and *at*, or TF-IDF weighting (which down weights uninformative words occurring across many documents). However, all such approaches have a fundamental limitation: they cannot recognize similarity between documents that do not have common words, like *Biden spoke in Nashville* and *The president greeted the press in Tennessee*.

To address this issue, various word embeddings have been developed (e.g., word2vec, GloVe, fastText), which use patterns of word usage in large corpora to derive fixed-length, real-valued vectors for words such that semantically similar words have similar vectors. Because *Biden* and *President* or *Nashville* and *Tennessee* will have similar vectors, the similarity of the above two sentences can be captured in various ways, e.g., by summing or averaging words in a sentence or document. Averaging or summing word embeddings (or even BoW) can be an excellent baseline in many document similarity tasks, but it also has a crucial shortcoming: simple averages or sums ignore word order and syntax. Thus, syntax and word order are tremendously important, cf. *Dog bites man* and *Man bites dog.*

This shortcoming, in turn, led to various deep neural network (DNN) approaches that exploit word embeddings but also respect the sequential nature of language. Recurrent neural networks like RNN and LSTM, for example, process text one token at a time, providing a nonlinear update to the current context based on the combination of new and old inputs. Convolutional neural networks (CNN), on the other hand, use so-called *convolution filters* (CFs) that are traditionally used in signal processing. The general idea is to learn multiple CFs which can capture useful, local features from a document. Recent transformer models like BERT and sentence-BERT are purely feedforward (dispensing with recurrence) but use positional embeddings and many self-attention layers to allow an order-sensitive document representation to emerge. These networks can be used in various ways to predict document similarity and generally form the current state-of-the-art in automatic measures of document similarity or duplication [11–14].

Any of the above types of neural networks can be used to create a Siamese neural network (SNNs), a class of neural network architectures which contains **two or more identical subnetworks**, each with the same configuration and parameters. Each subnetwork processes an input (here, a document), and the outputs of these subnetworks can then be compared to produce a prediction of similarity (Fig. 1a). SNNs have shown utility in document similarity, as well as signature verification [15–17] face recognition [18,19], and content-based (medical) image retrieval [20–22]. The advantage of SNNs are that they are more robust to class imbalance, can be easily combined with various downstream classifiers, and allow learning directly from gold labels of document similarity (instead of indirectly relevant information about, e.g., document classification). Siamese neural networks also offer more efficiency than does the most straightforward sentence similarity approach with BERT, which entails concatenating two documents (with a special [SEP] token between them), and passing the concatenation through BERT. Unfortunately, as this approach is quadratic in the number of documents, it quickly becomes intractable. Siamese neural networks, however, only require that each document be processed by the subnetwork once, which is only linear in the number of documents, and therefore much more efficient than the above BERT-based approach.

3. Dataset and pipeline

3.1. Corpus data set

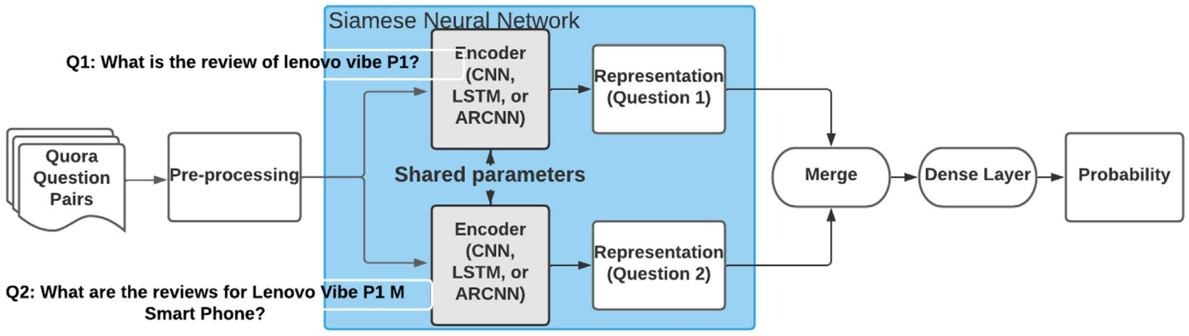
The Quora Question Pairs dataset contains a training set of 404,290 question pairs and a test set of 2,345,795 question pairs. This dataset is provided as part of a Kaggle shared-task competition. The training set has 255,027 (63.08%) pairs with a label 0 (not duplicate/similar) and 149,263 (36.92%) pairs with a label 1 (duplicate/similar). Those labels were manually determined by human experts. Unlike the training set, the publicly available test set does not provide labeled outcomes as a reference standard; thus, this study does not use the test set. As will be mentioned later, we instead conduct cross-validation on the training set. Each data entry consists of the following four data fields:

- id: unique ID of each pair
- qid1/qid2: ID for first/s question
- question1/question2: text of first/s questions
- is_dupliate: 0 indicates not duplicate, 1 indicates duplicate

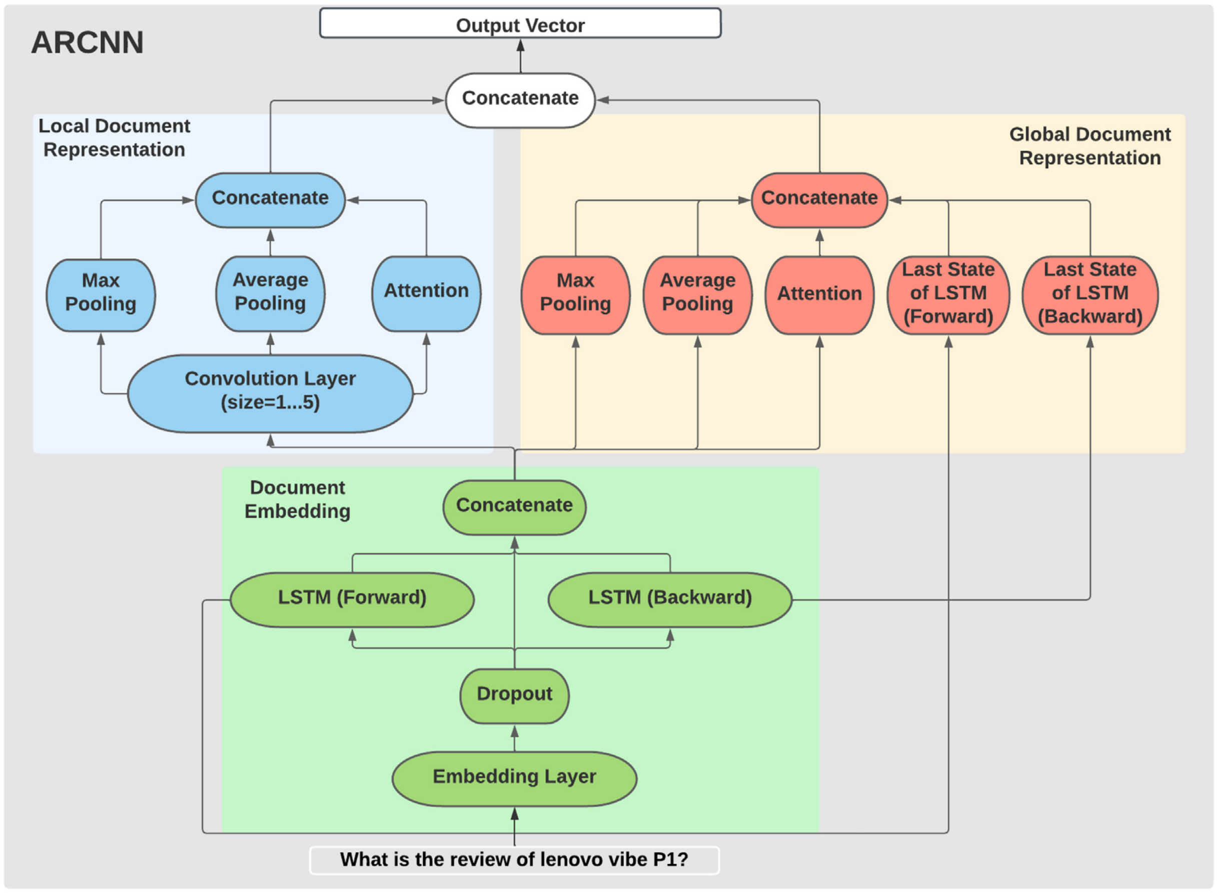
The question pairs cover a great range of topics including technology, entertainment, politics, culture, and philosophy. Some questions have special characters, such as math symbols and non-ASCII characters.

3.2. Text processing pipeline

Fig. 1a summarizes the pipeline shared by our novel S-ARCNN, as well as the S-LSTM and S-CNN, to which we compare. The pipeline first preprocesses the text of each question pair by (1) removing non-ASCII characters, (2) converting the contractions with ' to separate words (e.g., 't → not, 're → are, 'd → would, 'll → will, 've → have, i'm → i am), (3) adding space before and after punctuation, and (4) normalizing elongated words (lesssson → lesson). The pre-processed text of the question pair is then fed through a Siamese neural network (S-ARCNN, S-LSTM, or S-CNN), which yields a fixed length, real-valued vector representation for each question. These question representations are then merged, passed through an additional layer, and then passed to a final output node for determining the probability that the paired questions are duplicates.


Siamese Neural Network
Q1: What is the review of lenovo vibe P1?
Quora Question Pairs
Pre-processing
Encoder (CNN, LSTM, or ARCNN)
Shared parameters
Encoder (CNN, LSTM, or ARCNN)
Representation (Question 1)
Representation (Question 2)
Merge
Dense Layer
Probability
Q2: What are the reviews for Lenovo Vibe P1 M Smart Phone?

(a) Pipeline for processing document similarities with Siamese Neural Networks



(b) Attention-Augmented Recurrent Convolutional Neural Network

Fig. 1. Siamese Attention-Augmented Recurrent Convolutional Neural Network (S-ARCNN) Architecture, (b) is the represent the gray block: Encoder (ARCNN) in (a).

4. Models

4.1. Convolutional neural network

We compared our proposed ARCNN encoding strategy with CNN and LSTM approaches. The first encoding strategy is a CNN model. The input is the textual content of a question represented as a sequence of words $w = (w_1, w_2, \ldots, w_n)$ each represented by their corresponding index to the vocabulary V. The words are mapped to word vectors via an embedding matrix

$E \in \mathbb{R}^{|V| \times d}$ to produce a document matrix $D \in \mathbb{R}^{n \times d}$ where d is the dimension of the word representation vectors. The central idea is CNN employs a *convolution* operation over the document matrix to produce a feature map representation using a *convolution filter* (CF). We define a CF $W \in \mathbb{R}^{h \times d}$, where h is the window size, which represents the number of words we wish the convolution filter to span, and d is again the dimension of the word vectors. The 2-D convolution operation * is defined as

$$\mathbf{W} * D_{j:j+h-1} = \sum_{i=j}^{j+h-1} \sum_{k=0}^{d-1} \mathbf{W}_{i,k} D_{i,k}$$

where j is j-th row of document D. Next, we map a length h word window, $D_{j:j+h-1}$ of the document to a real number $c_j \in \mathbb{R}$ using a non-linear function f as in

$$c_j = f(\mathbf{W} * D_{i,j:j+h-1} + b)$$

where $b \in \mathbb{R}$ represents the bias term. After convolving over the entire document using $\mathbf{W}$, we have the corresponding convolved feature map

$$\mathbf{v} = [c_1, c_2, \cdots, c_{n-h+1}]$$

The goal is to learn multiple CFs that can collectively capture diverse representations of the same document. Choosing k filters results in k corresponding feature maps $\mathbf{v}^1, \cdots, \mathbf{v}^k$. The most distinctive feature of each feature map is selected using a max-over-time pooling operation [23] to produce the final feature vector $\mathbf{p} \in \mathbb{R}^k$, such that

$$\mathbf{p} = [v^1_{max}, \cdots, v^k_{max}]$$

where $v^j_{max} = max(\mathbf{v}^j_1, \cdots, \mathbf{v}^j_{n-h+1})$. Different sets of k CFs are typically learned for different window sizes, h. The window sizes are parameterized as a sequence $h_1, \cdots, h_H$ of H unique sizes. Suppose $\mathbf{p}^{(h_i)}$ denotes the feature vector produced on k filters with a window size of h_i, then the final $kH \times 1$ feature vector is

$$\mathbf{p}^* = p^{(h_1)} || \cdots || p^{(h_H)}$$

where $||$ is the vector concatenation operation. The resulting vector $\mathbf{p}^*$ provides a semantic representation of the question text. See previous work for additional details of CNNs used for text classification [24–26].

4.2. Long short-term memory

The second encoding strategy is an LSTM model. LSTM's are extensions of the basic RNN setup and were designed to obviate the vanishing gradient issue that plagued vanilla RNNs. LSTMs use Eq. 1 to decide how much information needs to be remembered from the previous output.

$$f_t = \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + b_f) \tag{1}$$

where f_t is forget gate, the hidden vector $\mathbf{h}_{t-1} \in \mathbb{R}^{a \times b}$ where a is the hidden dimension and b is 1, the input vector $\mathbf{x}_t \in \mathbb{R}^{c \times b}$ where c is the input embedding dimension and $\mathbf{W}_f \in \mathbb{R}^{d \times e}$ is the parameter matrix for the forget gate, where d is batch size and e is the sum of hidden and input embedding dimensions, i.e., $e = a + c$, [] is a concatenation operation between the hidden vector and the input vector, and the bias of forget gate $\mathbf{b}_f \in \mathbb{R}^{d \times b}$.

In Eq. 2, the input gate i_t is similar to the forget gate, but instead controls how much information needs to be remembered from the input vector where $\mathbf{W}_i \in \mathbb{R}^{d \times e}$ and $\mathbf{W}_c \in \mathbb{R}^{d \times e}$. Then we apply the hyperbolic tangent function, *tanh*, on the input vector to get $\tilde{C}_t$, the current state.

$$\begin{aligned} i_t &= \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \\ \tilde{C}_t &= tanh(\mathbf{W}_c \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C) \end{aligned} \tag{2}$$

Now, we let the model learn the information based on the output from forget gate and input gate. If f_t is close to zero, most information from previous stages ought to be forgotten as shown in Eq. 3.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3}$$

The last step is to create output state o_t with $\mathbf{W}_o \in \mathbb{R}^{d \times e}$, and get the new hidden layer vector $\mathbf{h}_t \in \mathbb{R}^{a \times b}$

$$\begin{aligned} o_t &= \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \\ \mathbf{h}_t &= o_t * tanh(C_t) \end{aligned}$$

We employ a bidirectional LSTM, which uses one LSTM to process the input from the first to the last token and another LSTM to process the input from the last to the first token. Typically, in a bidirectional LSTM, h_t of the last token of the forward LSTM and h_t of the first token of the backwards LSTM are concatenated and used as the sentence representation.

4.3. Siamese-ARCNN (S-ARCNN) models

The third encoding strategy is a novel approach we propose, and is a hybrid of the first two methods, with the addition of attention mechanisms. Again, we combine these components as they are thought to have complementary advantages: convolutional captures local features, recurrence with LSTM cells captures global features, and attention helps networks focus on the most relevant tokens. Fig. 1b illustrates our proposed ARCNN model. The model can be split into three parts. The first part is applying a bidirectional LSTM with 256 units (128 units in each direction), which converts the word embeddings H_e into context-aware word embeddings H_b (as shown in the green part).

The following two parts focus on the local and global levels of feature representation. First, we discuss the global document representation. In this module, we applied average pooling and max-over-time pooling on the H_b to get the representation $g_{ave} = ave(H_b)$ and $g_{max} = max(H_b)$, respectively. Second, the attention mechanism shown in Eq. 4 produces a document representation from the words by weighting the importance of each constituent word to the prediction task.

$$\begin{aligned} u_i &= h_{(c_i)} w_a, \quad a_i = e^{(u_i)} / \sum_i^n e^{(u_i)} \\ a_c &= \sum_i^n a_i h_{(c_i)} \end{aligned} \tag{4}$$

Where a context-aware token embedding $h_{(c_i)}$ is fed through a one-layer multilayer perceptron with weight matrix $\mathbf{w}_a$ to obtain a hidden representation u_i. Next, a normalized importance weight for each token is obtained through the softmax function. The document representation $\mathbf{a}_c$ is then the a_i-weighted sum of all tokens' context-aware embeddings $\mathbf{h}_{(c_i)}$. Finally, we concatenate all the outputs to have a global representation $\mathbf{p}^{global} = g_{ave}||g_{max}||LSTM_F||LSTM_B||a_{(c_{global})}$ where $LSTM_F$ and $LSTM_B$ represent the last state of the forward and backward LSTM's, respectively.

To calculate the local document representation, the LSTM outputs H_b pass through a convolutional layer to capture 1- to 5-gram features, $\mathbf{v}^1, \cdots, \mathbf{v}^5$. Max-over-time pooling and average pooling are then applied to obtain $\mathbf{l}_{max} = [v^1_{max}, \cdots, v^5_{max}]$ and $\mathbf{l}_{ave} = [v^1_{ave}, \cdots, v^5_{ave}]$, respectively. As in the global document representation, we also apply an attention mechanism to obtain $\mathbf{a}_{(c_{local})}$. The output for the local document representation is then $\mathbf{p}^{local} = l_{ave}||l_{max}||a_{(c_{local})}$. Finally, we concatenate the global and local representations to have the output of the encoding $\mathbf{p}^* = p^{global}||p^{local}$, which is the representation of a single document or sentence.

4.4. Baseline: logistic regression

As a baseline non-DNN model, we also considered a logistic regression (LR) on relatively simple features extracted from question pairs. In particular, the LR model uses a logistic function to compute the probabilities of a certain class as shown in Eq. 5

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

where $f(x) \in [0, 1]$ is the probability estimate of two questions being duplicates, x denotes the input values to the function, and e is the natural logarithm.

In this approach, in addition to the text pre-processing applied for the DNNs (see Section 3.2), we removed stopwords with the NTLK list [27]. We then constructed the following features: (1) the length of the question, (2) number of word types in common between the question pair, (3) the total number of words in the question pair, (4) if the last word of both questions is the same or not, (5) if the first word of both questions is same or not, (6) the average token length of both questions, (7) the absolute difference of the questions' lengths, (8 and 9) the number of common word types divided by the token length of each question, (10 and 11) the number of stopwords divided by the token length of each question, and (12 and 13) the number of non-stopword tokens divided by the token length of each question. Finally, we used the TF-IDF vectorizer from scikit-learn [28] to extract TF-IDF weights for each question. We then used these, as well as GloVe [29] vectors from spaCy [30], to calculate weighted averages of the word embeddings in a question, to obtain an embedding for each question. The 96-dimensional embeddings for each question were then concatenated to the 13 features mentioned above, so that there were 205 features in total for the logistic regression.

4.5. Transformers

As mentioned in the introduction, transformers (e.g., BERT [1]), have achieved state-of-the-art results in a wide variety of NLP tasks. Thus, we also tested three transformer-based models using Siamese encoding and cross-encoding (see Fig. 2) through Sentence-Transformers, a Python library for state-of-the-art sentence, text, and image embeddings [31] (for mathematical details of transformers, see [32]). Under both cross-encoding and Siamese encoding (referred to as bi-encoding in Sentence-Transformers), transformers are pre-trained with very large corpora (billions of tokens) on tasks like masked word prediction or next sentence prediction, and then fine-tuned via the cross-encoding or Siamese approach on tasks like detect-

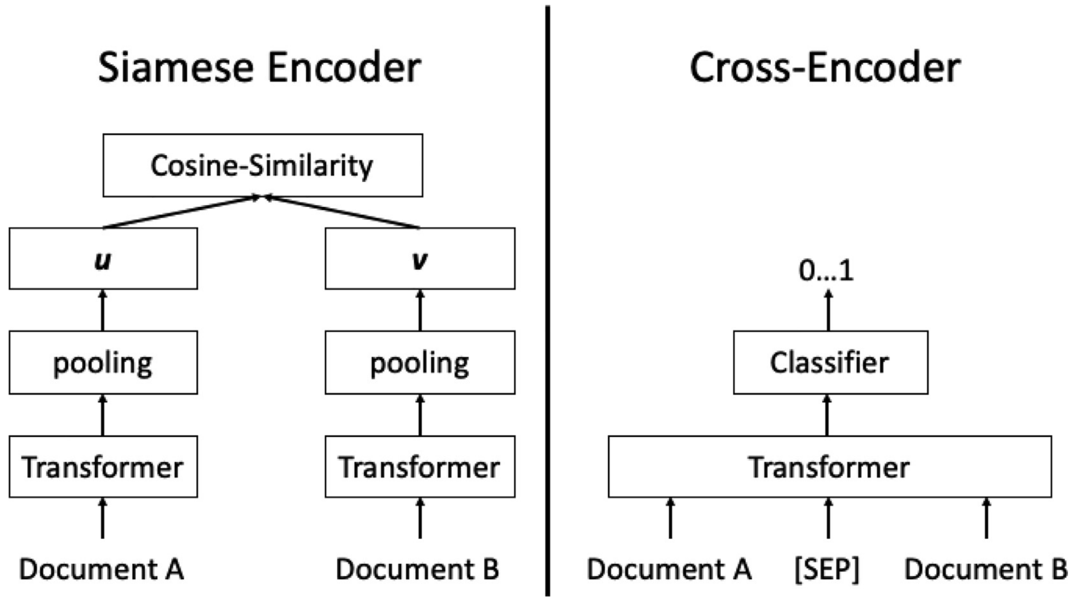


Fig. 2. Siamese vs cross-encoding in the Sentence-Transformers library. Figure adapted from [31].

ing duplicate questions or question–answer pairs. For fine-tuning under a cross-encoding approach, two documents are concatenated with the special [SEP] token between them, passed through a transformer, and then a classifier outputs the probability that the documents are duplicates, or an appropriate question–answer pair, etc. Under the Siamese approach, however, each document is processed by identical transformers, the final layer outputs are pooled to produce vectors $\mathbf{u}$ and $\mathbf{v}$ representing the first and second document, respectively, and then cosine similarity (or other metrics) between $\mathbf{u}$ and $\mathbf{v}$ can be calculated.

We tested a single pre-trained cross-encoder, *stsb-roberta-large*, which we refer to henceforth simply as CrossEncoder. We also tested two pre-trained Siamese encoders, *paraphrase-multilingual-mpnet-base-v2* and *all-mpnet-base-v2*. This latter model's (*all-mpnet-base-v2* fine-tuning included Quora Question Pairs, and thus we may have data leakage when we evaluate it on QQP and this model preforms the best among all the pre-trained model provided by Hugging face. As this model is a Siamese transformer fine-tuned with QQP, we refer to it as S-Trans(with QQP), and refer to the other Siamese transformer as just S-Trans. For all three models, we passed two documents through the cross-encoder or Siamese encoder to obtain a document similarity score, which is a cosine similarity in the Siamese encoder and a probability in the cross-encoder. We fed this single score into a logistic regression to find an optimal decision threshold. Thus, we emphasize that, in contrast to the other deep neural networks, we are *not* fine-tuning these transformers' millions of parameters to the QQP duplicate detection task. In addition, these models take raw text, i.e., we do not apply the text pre-processing steps described above.

5. Experiment and results

5.1. Experiment

We used 10-fold cross-validation to train and evaluate the three DNNs (Siamese-CNN, Siamese-LSTM, and Siamese-ARCNN), one traditional machine learning approach (Logistic Regression), and the transformers. First, we split the dataset into ten stratified folds. Then a model was trained in nine folds and tested in the remaining fold. After repeating this process ten times, we calculated average metrics: accuracy, precision, recall, and F1 score.

5.2. Model configuration for deep learning

All DNNs (except transformers) used commonly selected 300-dimensional GloVe embeddings [29] pre-trained on the Common Crawl corpus with 840 billion tokens to convert tokenized texts of length n into an embedding layer in $\mathbb{R}^{n\times 300}$. These embeddings were further updated by training. The maximum sequence length was set to 60 tokens, such that shorter and longer questions were padded or trimmed, respectively. As is standard [24], our S-CNN used convolution filters spanning 3, 4, and 5 words, and each filter size had 100 filters for the CNN encoding. We used a dropout rate at 0.5, set the final dense layer dimension to 100 and the hidden dimension to 200, and used a batch size of 2048. We used Adam optimization and a cross-entropy loss function. Models trained for a maximum of 30 epochs, with early stopping if loss did not change dramatically for 10 epochs. Our experiments were performed on a machine with Intel Xeon(R) Silver 4214 @ 2.20 GHz, 754 GB memory, and Tesla V-100 PCIE-32 GB graphical card. Our proposed S-ARCNN model's average training time was around 2 min for each epoch.

5.3. Evaluation measures

The precision, recall/sensitivity, and F1-score defined as follow:

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

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

$$F1 - score = 2 \ast \frac{Precision \ast \ Recall}{Precision + Recall}$$

where *TP*, *FP*, and *FN* are true positive, false positive, and false negative, respectively. The ideal classifier has *precision* and *recall* equal to one, which means *FP* and *FN* are zero. F1-score is the harmonic mean of *precision* and *recall.*

5.4. Results

Table 1 shows the accuracy, F1-score, precision, and recall with the corresponding confidence intervals. As can be seen, our Siamese Neural Network with CNN encoding has the best performance among the three deep learning models, our baseline model, and the five transformers. Because the transformers and baseline logistic regression did not perform competitively (F1's 5% or more below the three DNNs), we omit them from the following finer-grained analyses.

5.4.1. Secondary analysis results

In the Quora Question Pairs dataset, it turns out that most questions are, as one might guess, quite short. Fig. 3 shows the distribution of question lengths in number of words, and shows that the majority (64.44%) of questions have less than 12 words. In contrast, there were only 817 unique questions with more than 50 words.

Since the vast majority of questions are only of sentence or phrase length, our proposed S-ARCNN model may be too complicated for such short texts. Therefore, we hypothesized that our proposed model would perform better on long sentences. To test this, we calculated each model's accuracy among question pairs that have at least *N* words, for $N \in [1, 100]$. As Fig. 4 shows, performance of our proposed model, S-ARCNN, closes in on that of the S-CNN model as question length increases, becoming essentially identical with question pairs longer than 50 or 60 words.

Next, we tested our model on question pairs longer than 50 words. We found that as a question's length increases, the less likely it is a duplicate of another question. More specifically, when question pair length is $>$ = 50, only 624 out of 7273 pairs are duplicates, and when the length is $>$ = 80, only 13 out of 505 are duplicates. Similarly, in the full dataset, 37% of pairs are duplicates, while less than 3% are duplicates among question pairs over 80 words. Therefore, we believe it is more important to focus on the accuracy of the duplicate class (the minority class) than the overall accuracy. Among duplicates over 50 words, we found that S-ARCNN outperformed the Siamese based CNN model, as shown in Fig. 5.

Table 1
Model comparison for question pairs detection (95% CI).

	F1	Precision	Recall	Accuracy
LR[1]	62.06% (61.83%-62.28%)	64.80% (64.62%-64.99%)	59.53% (59.19%-59.88%)	73.12% (72.99%-73.25%)
CrossEncoder-Base[2]	73.60% (73.48%-73.71%)	60.06% (59.89%-60.22%)	95.04% (94.89%-95.19%)	74.83% (74.67%-74.99%)
CrossEncoder[3]	74.65% (74.49%-74.81%)	74.06% (73.88%-74.23%)	75.26% (74.92%-75.59%)	81.13% (81.03%-81.23%)
S-Trans-Base[4]	65.64% (65.58%-65.70%)	48.89% (48.83%-48.95%)	**99.86% (99.83%-99.88%)**	61.41% (61.31%-61.50%)
S-Trans[5]	72.80% (72.62%-72.98%)	70.39% (70.16%-70.61%)	75.39% (75.12%-75.65%)	79.20% (79.06%-79.35%)
S-Trans(with QQP)[6]	74.91% (74.69%-75.12%)	72.53% (72.28%-72.78%)	77.44% (77.15%-77.73%)	80.84% (80.68%-81.01%)
S-LSTM	79.62% (79.51%-79.74%)	79.26% (79.19%-79.34%)	80.80% (79.97%-80.43%)	80.69% (80.61%-80.77%)
S-ARCNN	79.83% (79.33%-80.33%)	80.35% (79.95%-80.74%)	79.50% (78.83%-80.17%)	81.46% (81.07%-81.84%)
S-CNN	**82.02% (81.83%-82.20%)**	**82.18% (81.99%-82.38%**	81.88% (81.60%-82.17%)	**83.32% (83.17%-83.47%)**

1 LR: Logistic Regression.
2 CrossEncoder: CrossEncoder with stsb-roberta-large pre-trained model without fed into logistic regression.
3 CrossEncoder: CrossEncoder with stsb-roberta-large pre-trained model.
4 S-Trans: Siamese Transformer with paraphrase-multilingual-mpnet-base-v2 pre-trained model without fed into logistic regression.
5 S-Trans: Siamese Transformer with paraphrase-multilingual-mpnet-base-v2 pre-trained model.
6 S-Trans(with QQP): Siamese Transformer with all-mpnet-base-v2 pretrained model.

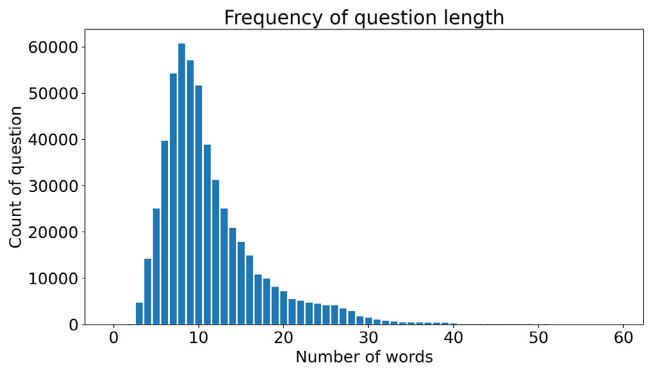


Fig. 3. Distribution of question lengths.

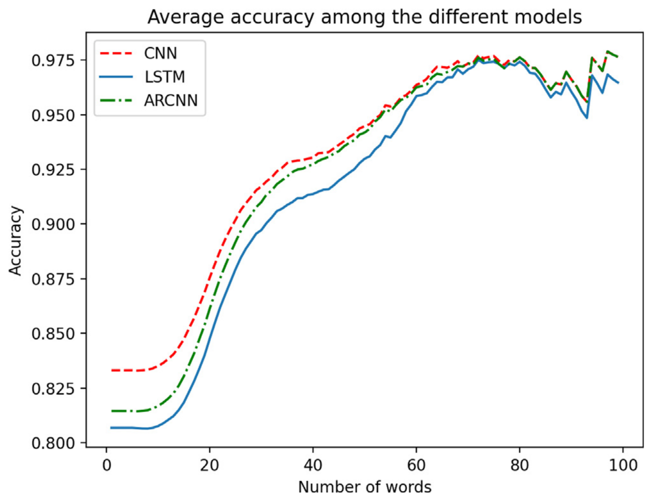


Fig. 4. Accuracy for different length of question pairs.

5.4.2. Error analysis

As another effort to better understand the S-ARCNN, we conducted an error analysis on the question pairs that contributed most to the loss. The three positive examples with the lowest predicted logits (false negatives) and three negative examples with the highest predicted logits (false positives) are shown in Table 2.

Reviewing the false positives, it is apparent that the questions are very similar, with just fairly subtle differences in certain concepts (FP1: *desktop* vs. *laptop*; FP2: *Colorado* vs. *Vermont*; FP3: *T795 44-in* vs. *T520 36-in*). This suggests that our model does not sufficiently distinguish subtle yet critical differences between concepts that are generally rather similar. One way to address this issue may be to train the model with an attention to the noun phrases to classify these two questions as the same or not. In the false negatives, FN1 and FN2 pairs have trouble with singular and plural *be* verbs. In the future, removing stopwords during pre-processing can handle this issue. In FN3, the adjectives *good* and *impressive* clearly mean the same things in these questions, yet our model predicts these questions are different. One way to address this is the same as handling the false positives, which is to force the model to pay attention to concepts (nouns, verbs, adjectives).

6. Discussion

In this study, we built a Siamese attention-augmented recurrent convolutional neural network (S-ARCNN) to score document pairs on their similarity, and evaluated it on the Quora Question Pairs dataset of duplicate questions. We also compared S-ARCNN to S-CNN, S-LSTM, S-ARCNN, and logistic regression models. Contrary to our expectations, a plain S-CNN (83% accuracy, 82% F1) outperformed our S-ARCNN model (81% accuracy, 80% F1), as well as S-LSTM (81% accuracy, 80% F1), our baseline logistic regression (73% accuracy, 62% F1), all with statistical significance at $p < .05$.

These DNN's also outperformed five models based on pre-trained transformers – *CrossEncoder-Base*, *CrossEncoder*, *S-Trans-Base*, *S-Trans*, and *S-Trans(with QQP)* – with the best of these (*S-Trans(with QQP)*) achieving 81% accuracy and 75% F1. We also found that fitting an optimal decision threshold for the transformer's embedding is helpful: the CrossEncoder with a fitted

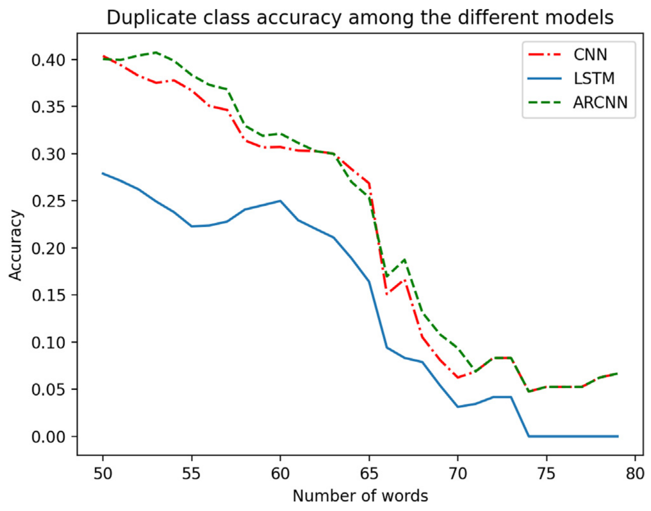


Fig. 5. Duplicate class accuracy for different length of question pairs.

Table 2
Error analysis for misclassified question pairs.

Error type	Pair	Question 1	Question 2
False positives	FP1	Which is the best desktop computer under 25000 INR?	What will be best laptop Under 25000 INR?
	FP2	What are the safety precautions on handling shotguns proposed by the NRA in Colorado?	What are the safety precautions on handling shotguns proposed by the NRA in Vermont?
	FP3	How does the HP OfficeJet 4620 Airprint compare to the HP DesignJet T795 44-in Printer?	How does the HP OfficeJet 4620 Airprint compare to the HP DesignJet T520 36-in Printer?
False negatives	FN1	Who is the bigger global competitor of bizbilla.com the global marketplace?	Who are the bigger global competitor of bizbilla.com the global marketplace?
	FN2	What is the review of lenovo vibe P1?	What are the reviews for Lenovo Vibe P1 M Smart Phone?
	FN3	What is a good inpatient drug and alcohol rehab center near Maricopa County Arizona?	What is the most impressive Inpatient Drug and Alcohol Rehab Center in Maricopa County Arizona?

decision threshold (*CrossEncoder*) improves the F1 score by 1% compare with *CrossEncoder-Base*. Similarly, the Siamese Encoder *S-Trans* outperformed the Siamese Encoder without a fitted threshold *S-Trans-Base* by about 7% in F1 score. These two results – poorer performance of pre-trained transformers compared to other DNN's, and superiority of pre-trained transformers with task-specific decision thresholds over pre-trained transformers without task-specific decision thresholds – both suggest that good performance relies on adapting the transformer for the task. Fitting a single decision threshold to the continuous scores output by the pre-trained transformers is a simple way to do this, but fine-tuning some or all of the transformers' internal layer weights to the task may improve performance further.

Unfortunately, it is difficult to compare our results to previous studies' results on QQP, as different studies have used different training and test sets, and it is not always known how these training and test sets were constructed (e.g., what positive–negative sample balance they used). For example, the original BERT paper [1] achieved an F1 score of 72.1% (with BERT-large) on the first Quora dataset release [33], with a training set of 364,000 samples and a test set of 391,000 samples. We, however, conducted 10-fold cross-validation on 255,027 samples, which implies training sets with 229,524 samples and test sets with 25,502 samples. Similarly, Yaghoobzadeh et al. [2] used BERT to embed a document, and fed these embeddings into a k-NN (k-Nearest Neighbor) model, and achieved an AUC of 80% with 9,100 training samples and a test of unknown size and positive–negative sample balance. Imtiaz et al. [4] used Siamese LSTM's with Manhattan distance for measuring similarity, and achieved 91.14% accuracy on QQP with a test set of 100,000 pairs of questions. Part of this problem in standard train-test splits is likely because, as we noted, the publicly available QQP test set does not contain labels. Instead, participants in the Kaggle competition needed to submit their test predictions to a website, which conducted evaluation for them, to minimize test set leakage. Thus, researchers have chosen their own ad hoc splits of the Kaggle training set. Reaching a consensus throughout the field on shared train-test (or train-dev-test) splits of QQP is therefore important for researchers to make appropriate comparisons between each other's work.

We attribute the success of the S-CNN's performance to the fact that the vast majority of questions are short (90% of questions have fewer than 20 words). Thus, a simple model that mostly extracts local features (as CNN's are typically thought to do [6]) could be enough to represent the meanings of most questions. Consistent with this hypothesis, in the secondary analysis, we found that when the length of the questions pairs was greater than 50 words, our S-ARCNN model performed better than the S-CNN model for identifying duplicate questions. We speculate that this is because our S-ARCNN model has an LSTM component to extract global features from longer documents [6].

As shown in Fig. 4, there is a drop in accuracy for questions between 80 and 95 words long. This is because the longer question pairs are less likely to be duplicates; as the length increases, the percentage of duplicate questions among all question pairs is changing, and the misclassification on the duplicate class affects the accuracy.

We also found that when word length increased, accuracy increased. One simple explanation for this is that the longer the question, the less likely it has a duplicate. Therefore, the overwhelming majority class in the long question pairs is non-duplicate. However, we also suspect that longer questions contain more information for the model to make a decision, therefore increasing the chance that a model makes a correct prediction.

Lastly, an error analysis revealed that our model seemed not to pay appropriate attention to subtle differences between concepts (e.g., *Vermont* vs *Colorado*). This error analysis provides a guideline for us to further improve our model, e.g., forcing a model to pay extra attention to nouns and other concepts.

6.1. Limitations

This study contains the following limitations. First, the dataset skewed to short questions: 99.8% of questions have less than 50 words, and 64.4% of questions have less than 12 words. Relatedly, duplicates are increasingly rare among longer question pairs: among the 7273 pairs with $>$ 50 words, 624 (8.58%) are duplicates, and among the 505 pairs with $>$80 words, only 13 (2.57%) are duplicates. With such a small number of longer questions, and with such skew in the label distribution, models might not be able to effectively learn what duplicates look like among longer documents. We are therefore cautious in drawing conclusions about the S-ARCNN model's performance across the full range of possible document lengths. Second, relatedly, the Quora Question Pairs, while covering a broad range of subjects, still represent a very particular type of text (questions). To address these limitations, we plan to reproduce our tests on longer documents of other types (see next section). Last, there are no available annotation guidelines for this QQP public dataset. The ground truth labels are inherently subjective, as the true meaning of sentences can never be known with certainty. Human labeling is also a 'noisy' process, and reasonable people will disagree. Therefore, the dataset labels are not 100% accurate, and may include incorrect labeling. The effect of incorrect labeling on our model is unknown.

7. Conclusions and future work

State-of-the-art approach in automatically measuring document similarity involves deep neural networks with attention, recurrence, convolution, and Siamese networks. These components are thought to serve complementary functions, yet their combination in a single architecture is underexplored. We therefore developed and evaluated an attention-augmented recurrent convolutional Siamese neural network for matching duplicate questions. Overall, the simple Siamese-CNN model performed best, given that most of the questions contained fewer than 12 words. However, our model performed best for the duplicate question pairs with longer sentences, i.e., a total length of more than 50 words. Given this apparent advantage of S-ARCNN in matching longer documents, we plan to apply our proposed model to match longer documents such as scientific article abstracts. This could help researchers identify collaborators with similar research interests, or help editors find potential reviewers, or match resumes with job descriptions. Additionally, comparing the similarity between a special request for proposal (RFP) and a researcher's biosketch containing a brief introduction on the researcher's interests can help researchers find matched RFPs.

CRediT authorship contribution statement

Sifei Han: Investigation, Methodology, Software, Visualization, Writing - original draft. **Lingyun Shi:** Formal analysis, Validation, Writing - review & editing. **Russell Richie:** Formal analysis, Investigation, Writing - review & editing. **Fuchiang R. Tsui:** Conceptualization, Investigation, Methodology, Supervision, Visualization, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 2031150 and Children's Hospital of Philadelphia. The content is solely the responsibility of the authors and does not necessarily represent the official views of funding agencies. We would like to gratefully acknowledge Edward Gruver from the Tsui Lab at the Children's Hospital of Philadelphia and anonymous reviewers for their valuable feedback.

Appendix A. Training Deep Models and Pseudocode

We train our proposed model using a binary cross-entropy loss

$$Loss = -\sum_{i=1}^{L} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

where $\hat{y}_i$ is the i-th scalar value in the model output, $y_i \in \{0, 1\}$ is the ground truth for the i-th label.

Algorithm 1: Pseudocode for S-ARCNN Model

```
1  S-ARCNN $(q1, q2, y)$;
   Input  : Question pairs $q1$ and $q2$, with label $y \in \{0, 1\}$
   Output: $0$ : not duplicate, $1$ : duplicate
2  Initialize $W_i$, $A_i$(word embedding, attention embedding)
3  while not convergent do
4      for $i \in \{0, \ldots, N\}$ do
5          $x_1$ = embed(q1)
6          $x_2$ = embed(q2)
7          $x_1$ = BiLSTM($x_1$)
8          $x_2$ = BiLSTM($x_2$)
9          #Local Document Representation
10         $x_1\_cnn = CNN1(x_1) \| CNN2(x_1) \| \ldots \| CNN5(x_1)$
11         $x_2\_cnn = CNN1(x_2) \| CNN2(x_2) \| \ldots \| CNN5(x_2)$
12         $x_1\_cnn =$
             $Max\_pool(x_1\_cnn) \| Average\_pool(x_1\_cnn) \| Attention(x_1\_cnn)$
13         $x_2\_cnn =$
             $Max\_pool(x_2\_cnn) \| Average\_pool(x_2\_cnn) \| Attention(x_2\_cnn)$
14         #Global Document Representation
15         $x_1\_lstm =$
             $Max\_pool(x_1) \| Average\_pool(x_1) \| Attention(x_1) \| \overrightarrow{x_1} \| \overleftarrow{x_1}$
16         $x_2\_lstm =$
             $Max\_pool(x_2) \| Average\_pool(x_2) \| Attention(x_2) \| \overrightarrow{x_2} \| \overleftarrow{x_2}$
17         $x_1 = x_1\_cnn \| x_1\_lstm$
18         $x_2 = x_2\_cnn \| x_2\_lstm$
19         $\hat{y}$ = sigmoid($x_1 \| x_2$)
20     end
21     Calculate binary cross-entropy $\mathbf{J}(\theta)$
22     loss = reduce_mean($\mathbf{J}(\theta)$)
23     update the network parameters basis on loss by using back
         propagation $\theta_{t+1} = \theta_t - \alpha \frac{\partial E(X, \theta_t)}{\partial \theta}$
24 end
```

References

[1] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423.https://aclanthology.org/N19-1423.

[2] Y. Yaghoobzadeh, A. Rochette, T.J. Hazen, Cross-domain generalization through memorization: A study of nearest neighbors in neural duplicate question detection, arXiv preprint arXiv:2011.11090 (2020).

[3] C. McCreery, N. Katariya, A. Kannan, M. Chablani, X. Amatriain, Domain-relevant embeddings for medical question similarity, arXiv preprint arXiv:1910.04192 (2019).

[4] Z. Imtiaz, M. Umer, M. Ahmad, S. Ullah, G.S. Choi, A. Mehmood, Duplicate questions pair detection using siamese malstm, IEEE Access 8 (2020) 21932–21942.

[5] T. Zheng, Y. Gao, F. Wang, C. Fan, X. Fu, M. Li, Y. Zhang, S. Zhang, H. Ma, Detection of medical text semantic similarity based on convolutional neural network, BMC Med. Inform. Decision Making 19 (1) (2019) 1–11.

[6] W. Yin, K. Kann, M. Yu, H. Schütze, Comparative study of cnn and rnn for natural language processing, arXiv preprint arXiv:1702.01923 (2017).

[7] D. Huang, A. Ahmed, S.Y. Arafat, K.I. Rashid, Q. Abbas, F. Ren, Sentence-embedding and similarity via hybrid bidirectional-lstm and cnn utilizing weighted-pooling attention, IEICE Trans. Inform. Syst. 103 (10) (2020) 2216–2227.

[8] B. Li, L. Han, Distance weighted cosine similarity measure for text classification, in: International conference on intelligent data engineering and automated learning, Springer, 2013, pp. 611–618.
[9] M. Nishom, Perbandingan akurasi euclidean distance, minkowski distance, dan manhattan distance pada algoritma k-means clustering berbasis chi-square, J. Inform. 4 (01) (2019).
[10] S. Niwattanakul, J. Singthongchai, E. Naenudorn, S. Wanapu, Using of jaccard coefficient for keywords similarity, in: Proceedings of the international multiconference of engineers and computer scientists, vol. 1, 2013, pp. 380–384.
[11] O. Khattab, M. Zaharia, Colbert: Efficient and effective passage search via contextualized late interaction over bert, in: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, 2020, pp. 39–48.
[12] N. Peinelt, D. Nguyen, M. Liakata, tbert: Topic models and bert joining forces for semantic similarity detection, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7047–7055.
[13] J. Rabelo, M.-Y. Kim, R. Goebel, Combining similarity and transformer methods for case law entailment, in: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law, 2019, pp. 290–296.
[14] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, arXiv preprint arXiv:1908.10084 (2019).
[15] S. Dey, A. Dutta, J.I. Toledo, S.K. Ghosh, J. Lladós, U. Pal, Signet: Convolutional siamese network for writer independent offline signature verification, arXiv preprint arXiv:1707.02131 (2017).
[16] Z.-J. Xing, F. Yin, Y.-C. Wu, C.-L. Liu, Offline signature verification using convolution siamese network, in: Ninth International Conference on Graphic and Image Processing (ICGIP 2017), Vol. 10615, International Society for Optics and Photonics, 2018, p. 106151I.
[17] J. Bromley, J.W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, R. Shah, Signature verification using a 'siamese' time delay neural network, Int. J. Pattern Recognit Artif Intell. 7 (04) (1993) 669–688.
[18] Y. Taigman, M. Yang, M. Ranzato, L. Wolf, Deepface: Closing the gap to human-level performance in face verification, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 1701–1708.
[19] L. Song, D. Gong, Z. Li, C. Liu, W. Liu, Occlusion robust face recognition based on mask learning with pairwise differential siamese network, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 773–782.
[20] Ş. Öztürk, Two-stage sequential losses based automatic hash code generation using siamese network, Avrupa Bilim ve Teknoloji Dergisi (2020) 39–46.
[21] Ş. Öztürk, A. Alhudhaif, K. Polat, Attention-based end-to-end cnn framework for content-based x-ray image retrieval, Turkish J. Electr. Eng. Comput. Sci. 29 (SI-1) (2021) 2680–2693.
[22] Ş. Öztürk, Hash code generation using deep feature selection guided siamese network for content-based medical image retrieval, Gazi Univ. J. Sci. (2021), 1–1.
[23] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, J. Mach. Learn. Res. 12 (Aug) (2011) 2493–2537.
[24] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1746–1751. doi:10.3115/v1/D14-1181.https://aclanthology.org/D14-1181.
[25] A. Rios, R. Kavuluru, Convolutional neural networks for biomedical text classification: application in indexing biomedical articles, in: Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics ACM, 2015, pp. 258–267.
[26] S. Han, T. Tran, A. Rios, R. Kavuluru, Team uknlp: Detecting adrs, classifying medication intake messages, and normalizing adr mentions on twitter., in: SMM4H@ AMIA, 2017, pp. 49–53.
[27] E. Loper, S. Bird, Nltk: The natural language toolkit, arXiv preprint cs/0205028, 2002.
[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (Oct) (2011) 2825–2830.
[29] J. Pennington, R. Socher, C.D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.
[30] M. Honnibal, I. Montani, spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing, to appear (2017).
[31] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019.
[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inform. Process. Syst. 30 (2017).
[33] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, S.R. Bowman, Glue: A multi-task benchmark and analysis platform for natural language understanding, arXiv preprint arXiv:1804.07461 (2018).