EmBoost: Embedding Boosting to Learn Multilevel Abstract Text Representation for Document Retrieval

Tolgahan Cakaloglu[1,2], Xiaowei Xu[2] and Roshith Raghavan[3]
[1]*Walmart Labs, Dallas, Texas, U.S.A.*
[2]*University of Arkansas, Little Rock, Arkansas, U.S.A.*
[3]*Walmart Labs, Bentonville, Arkansas, U.S.A.*

Keywords: Natural Language Processing, Information Retrieval, Deep Learning, Learning Representations, Text Matching.

Abstract: Learning hierarchical representation has been vital in natural language processing and information retrieval. With recent advances, the importance of learning the context of words has been underscored. In this paper we propose EmBoost i.e. Embedding Boosting of word or document vector representations that have been learned from multiple embedding models. The advantage of this approach is that this higher order word embedding represents documents at multiple levels of abstraction. The performance gain from this approach has been demonstrated by comparing with various existing text embedding strategies on retrieval and semantic similarity tasks using Stanford Question Answering Dataset (SQuAD), and Question Answering by Search And Reading (QUASAR). The multilevel abstract word embedding is consistently superior to existing solo strategies including Glove, FastText, ELMo and BERT-based models. Our study shows that further gains can be made when a deep residual neural model is specifically trained for document retrieval.

1 INTRODUCTION

The objective of question answering system (QA) is to generate concise answers to arbitrary questions asked in natural language. Given the recent successes of increasingly sophisticated neural attention based question answering models, (Yu et al., 2018), the QA task can be broken into two as suggested by (Chen et al., 2017), (Cakaloglu and Xu, 2019):

- Document retrieval: Retrieval of the document most likely to contain all the information to answer the question correctly.
- Answer extraction: Utilizing one of the above question- answering models to extract the answer to the question from the retrieved document.

We used a collection of unstructured natural language documents as knowledge base to retrieve answers for questions. In this study we aim to explore and compare the quality of various retrieval strategies and further investigate the feasibility of training specialized neural network models for document retrieval. Towards this end embedding strategies play a pivotal role in retrieval by converting the text to vector representation. Traditional word embedding methods learn hierarchical representations of documents where each layer gives a representation that is a high-level abstraction of the representation from a previous layer. Most text embedding strategies utilize either the highest layer like Word2Vec (Mikolov et al., 2013), or an aggregated representation from the last few layers, like in ELMo (Peters et al., 2018) to generate representation for information retrieval.

In this paper, we present a new text embedding strategy called *EmBoost* that consists of two steps as shown in Figure 1. In the first step, a mixture of weighted representations across the entire hierarchy of text embedding model is formed so as to preserve all levels of abstraction. In the second step, all representations from various models are combined to generate an ensemble representation and used for document retrieval task. This strategy takes advantage of the abstraction capabilities of individual embedding models and various models complement one another to create higher quality embedding. Taking the example of *"⋯ match ⋯"* in Figure 1, different levels of representation of *"match"* including word level (word sense) and concept level (abstract meaning like competition, resemblance, and burning wick) are aggregated to form a mixture of representations. In the second step, all these mixture representations from different word embedding models are aggregated

Cakaloglu, T., Xu, X. and Raghavan, R.
EmBoost: Embedding Boosting to Learn Multilevel Abstract Text Representation for Document Retrieval.
DOI: 10.5220/0010822900003116
In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 3*, pages 352-360
ISBN: 978-989-758-547-0; ISSN: 2184-433X

to form an ensemble representation, which takes advantage of the complementary strength of individual models and corpora. Consequently, EmBoost delivers the power of multilevel abstraction with the strength of individual models.
Further in this study we introduce a convolutional residual network (ConvRR) over the embedding vectors to improve the performance of document retrieval task. The retrieval performance is further improved by employing triplet learning with (semi-)hard negative mining on the target corpus i.e. add a margin to the positive sample such that the negative sample is closer to the anchor thereby forcing the model to learn to solve such hard cases: $||\boldsymbol{q}_{\text{anchor}}, \boldsymbol{d}_{\text{positive}}|| < ||\boldsymbol{q}_{\text{anchor}}, \boldsymbol{d}_{\text{negative}}|| < ||\boldsymbol{q}_{\text{anchor}}, \boldsymbol{d}_{\text{positive}}|| + margin$

This paper is structured as follows: First, we review recent advances in text embedding in Section 2. In Section 3 we dive deeper into the details of our approach. More specifically, we describe EmBoost approach followed by a formulation of deep residual retrieval model that can be used to augment the text embedding and thereby enhance the quality of document retrieval. In this paper we compare the proposed method to the baselines that utilize existing popular text embedding models. Experimentation details such as datasets, evaluation metrics and implementation details can be found in Section 4. The results are reported in Section 5. Future work in Section 6 discusses potential improvements and spin off studies.

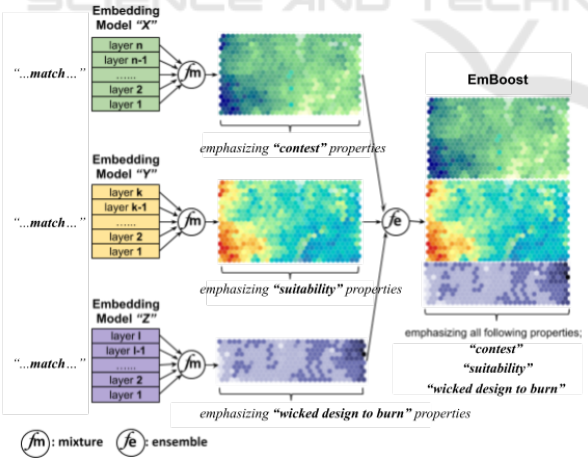

Figure 1: The illustration of EmBoost method using an example of "$\cdots$ *match* $\cdots$"

2 RELATED WORK

One of the most widely used measure for ranking importance of a word or token in a document is the term frequency-inverse document frequency or TF-IDF as proposed by (Salton and McGill, 1986). TF-IDF calculates a weighting factor for each token and is widely employed in information retrieval and text mining. Significant recent advances have been made in word embedding which is the way to convert text to numeric vectors. The literature on embedding strategies is extensively covered by (Perone et al., 2018). Word2Vec by (Mikolov et al., 2013), which is built upon on the neural language model for distributed word representations by (Bengio et al., 2003), has become widely adopted in natural language processing. It is a shallow network that can conserve semantic relationships between words and their context; or in other terms, surrounding words. The two approaches proposed in Word2Vec are the Skip-gram model which predicts surrounding words from the target word and the Continuous Bag-of-Words (CBOW) which predicts target word given the surrounding words. Global Vectors (GloVe) by (Pennington et al., 2014), was proposed to address some of the limitations of Word2Vec by focusing on the global context instead of the immediate surrounding words for learning the representations. The global context is calculated by utilizing the word pair co-occurrences in a corpus. During this calculation, a count-based approach is performed, unlike the prediction-based method in Word2Vec.

Another very popular embedding approach has been fastText, by (Mikolov et al., 2018). Conceptualy Word2Vec and Fasttext work in a similar fashion to learn vector representations of words. But unlike Word2Vec, which uses words to predict words, fastText treats each word as being composed of character n-grams. The vector for a word is made of the sum of this character n-grams. To further extract high quality meaningful representation embedding from Language Models (ELMo) was proposed by (Peters et al., 2018). ELMo extracts representations from a bi-directional Long Short Term Memory (LSTM), (Hochreiter and Schmidhuber, 1997), that is trained with a language model (LM) objective on a very large text corpus. ELMo representations are a function of the internal layers of the bi-directional Language Model (biLM) that outputs good and diverse representations about the words/token (a convolutional neural network over characters). ELMo is also incorporating character n-grams, as in fastText, but there are some constitutional differences between ELMo and its predecessors.

The encoder-decoder approaches however function by compressesing the input source sentence into a fixed length vector. This has shown to lead to decline in performance when dealing with long sentences. Additionally, the sequential nature of the model architecture prevents parallelization. To overcome these challenges, attention based transformer architecture

was proposed (Bahdanau et al., 2016; Vaswani et al., 2017).

One of the earliest Transformer based model to come out was BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) which was pre-trained on a large corpus of unlabeled text including entire Wikipedia (2.5 billion words) and Book corpus (800 million words). A key takeaway about BERT is that it is a deeply bidirectional model allowing it to learn from both the left and right side of a token's context during the training phase. This bidirectionality is important for truly understanding the meaning of language. BERT has been optimized further with XLNet (Yang et al., 2020) and RoBERTa (Liu et al., 2019) whereas DistilBERT (Sanh et al., 2020) improves on the inference speed. This was then followed by sentence-BERT (Reimers and Gurevych, 2019) which adapted the BERT architecture by using siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity.

Last, but not least, distance metric learning is a technique to learn a distance metric for invariant data representations in a way that retains the related vectors close to each other while separating different ones in the vector space, as stated by (Lowe, 1995), (Cao et al., 2013), and (Xing et al., 2002). However, instead of using standard distance metric learning, using deep networks to infer a non-linear embedding of data has shown significant improvements when it comes to learning representations using various loss functions, including triplet loss—by (Hadsell et al., 2006), (Chopra et al., 2005)—, contrastive loss—by (Weinberger and Saul, 2009), (Chechik et al., 2010)—, angular loss—by (Wang et al., 2017)—, and n-pair loss—by (Sohn, 2016)—for influential studies—by (Taigman et al., 2014), (Sun et al., 2014), (Schroff et al., 2015), and (Wang et al., 2014)—.

After providing a brief review of the latest trends in the field, we describe the details of our approach and experimental results in the following sections.

3 PROPOSED APPROACH

3.1 Overview

The proposed approach for document retrieval begins with first devising a new text embedding approach called EmBoost which is an ensemble of multilevel abstract representations learned from multiple distinct pre-trained text embedding models. Secondly a neural network model called **ConvRR** ,short for Convolutional Residual Retrieval Network (and alternatively **FCRR**, short for Full-Connected Retrieval Network by (Cakaloglu et al., 2018), is trained using triplet loss. The general architecture of the proposed ConvRR model is shown in Figure 2.

The model begins with a series of word inputs $w_1, w_2, w_3, ..., w_k$, that could represent a phrase, sentence, or paragraph. Those inputs, then, are initialized with different resolutions of pre-trained embedding models which may be context-free, contextual or numerical. ConvRR then further improves the multilevel abstract representation by using convolutional blocks through residual connection to the initialized original embedding. The residual connection enables the model not to lose the meaning and the knowledge derived from the pre-trained multilevel abstract embedding and enables it to make some adjustments to its knowledge using limited additional training data. A final representation is then sent to the retrieval task.

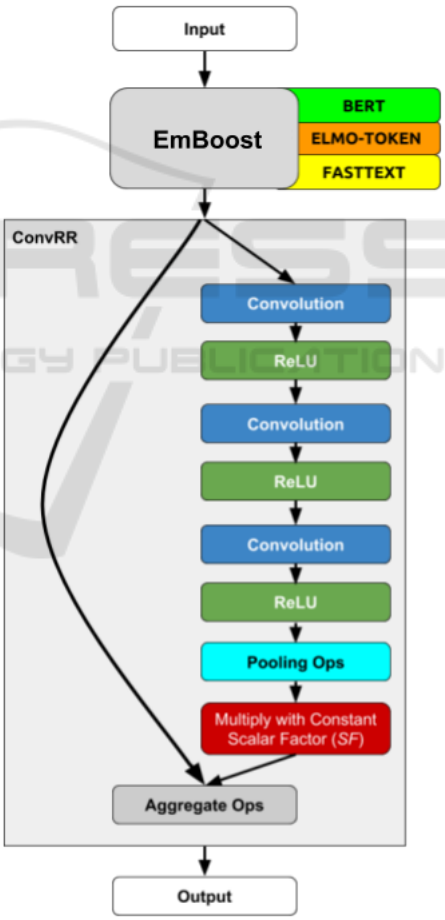

Figure 2: An overview of proposed approach consisting of an ensemble of multilevel abstract representations learned from multiple distinct pre-trained text embedding models and the *Convolutional Residual Retrieval Network* (**ConvRR**).

3.2 EmBoost

Existing powerful pre-trained text embedding strategies are trained using different *data sources* (Wikipedia, Common Craw, and etc.) as well as different *techniques* (supervised, unsupervised or variations). These pre-trained representations can broadly belong to three types- *context-free* (GloVe, FastText, etc.), or *contextual* (ELMo, Bert, etc.), and *statistical* (term frequency–inverse document frequency). The contextual strategies can further be *unidirectional* or *bidirectional*. Generally, context-free and statistical text embeddings are represented as a vector whereas contextual text embedding strategies are represented as a matrix.

Traditionally, one of pre-trained embedding models is selected to initialize a network for a defined downstream task. Hence, a series of word inputs to the network is initialized using the selected embedding model. If the selected embedding model generates a matrix instead of d-dimensional vector, then the matrix for each word is represented as follows:

$$\mathbf{E}_i^j = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_l]_{l \times d} \tag{1}$$

where $\mathbf{E}_i^j \in \mathbb{R}^{l \times d}$ is the $l \times d$-dimensional pre-trained word matrix of i-th word input, j denotes the given embedding model and l represents the number of layers in the embedding model. Averaging all the layers (ELMo), $\frac{1}{l}\sum_{i=1}^{l} \mathbf{e}_i$, or concatenating each of the last 4 layers (Bert), $< \mathbf{e}_{l-3} \oplus \mathbf{e}_{l-2} \oplus \mathbf{e}_{l-1} \oplus \mathbf{e}_l >$ in the matrix are the best practice to create a d'-dimensional vector, where $d' = d$ if averaging all layers is used, and $d' = L \times d$ in case concatenating is used, L is the number of layers (We used the notation of L to refer to the selected layers from l-layers embedding model). Current practice is to use only the last layer or top few layers, while we propose to consider all layers for multilevel abstract representation.

The proposed multilevel abstract word embedding has two cascaded operations: $f_{mixture}(\cdot,\cdot,\cdot)$ and $f_{ensemble}(\cdot)$.

Forming a mixture of the representations from an embedding model, $f_{mixture}(\cdot,\cdot,\cdot)$, can be formulated as below:

$$\mathbf{x}_i^j = f_{mixture}(\mathbf{E}_i^j, w_{idf}, \mathbf{m}^j) \tag{2}$$

where $\mathbf{m}^j \in \mathbb{R}^l$ is a coefficient vector and $\sum_{i=1}^{l} m_i^j = 1$. Each coordinate of the $\mathbf{m}^j$ represents a magnitude to weight the corresponding layer of the model $\mathbf{E}_i^j$. w_{idf} denotes an IDF weight of the i-th word input. $f_{mixture}(\cdot,\cdot,\cdot)$ is an aggregate function, which aggregates the input using an operation such as *sum*, *average*, and *concatenate*. Weighted layers of the model $\mathbf{E}_i^j$ are then computed by that aggregate function. x_i^j is the d'-dimensional vector where $d' = d$ if $f_{mixture}(\cdot,\cdot,\cdot)$ is defined by *sum* or *average*, and $d' = d \times l$ in case $f_{mixture}(\cdot,\cdot,\cdot)$ is defined by *concatenate*. The obtained mixture of representations from multiple word embedding models can form an ensemble representation as follows.

$$X_i' = \{\mathbf{x}_i^1, \mathbf{x}_i^2, \cdots \mathbf{x}_i^n\} \tag{3}$$

where X_i' is a set of representations from different embedding models, using $f_{mixture}(\cdot,\cdot,\cdot)$ for the i-th word input and n is the number of embedding models. $f_{ensemble}(\cdot)$ is a function to aggregate all representations in X_i' and can be defined as follows:

$$\mathbf{x}_i = f_{ensemble}(X_i', \mathbf{u}) \tag{4}$$

where $f_{ensemble}(\cdot)$ is also a aggregate function defined by an operation like *sum*, *average*, and *concatenate*. Note that, representations are coerced to a common length, if $f_{ensemble}(\cdot)$ is defined by *sum* or *average*. Additionally, $\mathbf{u} \in \mathbb{R}^n$ is a coefficient vector and $\sum_{j=1}^{n} \frac{u^j}{||u||} = 1$. Each coordinate of the $\mathbf{u}$ represents a magnitude to weight the corresponding embedding model of the ensemble of text embedding models. Hence, $\mathbf{x}_i$ is d''-dimensional multilevel abstract word embedding of the i-th word input. The pseudo-code of the proposed approach is shown in Algorithm 1.

Algorithm 1: EmBoost for i-th word input.

Input:
idf weight: w_{idf}
set of embedding models: $\mathbf{E}_i = \{\mathbf{E}_i^1, \mathbf{E}_i^2, \cdots, \mathbf{E}_i^n\}$,
set of coefficient vectors, one for each model:
$\mathbf{M} = \{\mathbf{m}^1, \mathbf{m}^2, \cdots, \mathbf{m}^n\}$,
vector of coefficient values, one for each model:
$\mathbf{u} = [u^1, u^2, \cdots, u^n]$
Output:
EmBoost $\mathbf{x}_i$
Begin
$X_i' = \{\}$
for $j = 1$ **to** n **do**
  $\mathbf{x}_i^j = f_{mixture}(\mathbf{E}_i^j, w_{idf}, \mathbf{m}^j)$
  $\mathbf{x}_i^j$ is added to X_i'
end for
$\mathbf{x}_i = f_{ensemble}(X_i', \mathbf{u})$
Return: $\mathbf{x}_i$
End

With the ensemble text embedding approach, we are generating embedding by taking the following aspects into consideration:

- Multi-sources: Instead of relying on one pre-trained embedding model, we want to utilize the power of multiple pre-trained embedding models since they are trained using different data source as well as different techniques. Therefore, integrating

different word embedding models can harness the complementary power of individual models.

- Different layers: We take the embedding from different layers of $\mathbf{E}$ each embedding model instead of just the last layer or few top layers.
- Weighted embedding: Incorporating word embedding with a weighting factor like inverse document frequency (*IDF*) or bm25 produce better results for information retrieval and text classification as presented by (Boom et al., 2015).

  An IDF is formulated as: $log_e(\frac{\#ofdocuments}{df_w})$, where a document frequency (df_w) is the number of documents in the considered corpus that contain that particular word w.

3.3 ConvRR

To further improve the performance of document retrieval, a convolutional residual retrieval (ConvRR) model is trained on top of the proposed ensemble of text embeddings. The model is presented in Figure 2. Let $\mathbf{x}_i \in \mathbb{R}^{d''}$ be the d''-dimensional proposed multilevel abstract word embedding of the i-th word input in a text; therefore, the word inputs can be denoted as a matrix:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \cdots, \mathbf{x}_k]_{k \times d''} \tag{5}$$

where k is the number of word inputs in a text. The ConvRR generates feature representations, which can be expressed as the following:

$$\mathbf{X}'' = f(\mathbf{W}, \mathbf{X}, sf) \tag{6}$$

$$\mathbf{o} = \mathbf{X}'' + \frac{1}{k}\sum_{i=1}^{k} \mathbf{x}_i \tag{7}$$

where $f(\cdot,\cdot,\cdot)$ is the convolutional residual retrieval network that executes series of convolutional components (a convolution and a rectified linear unit (ReLU) (Nair and Hinton, 2010)), a pooling, and a scaling operation. $\mathbf{X}''$ is produced by multilevel abstract word embedding X with trainable weights $\mathbf{W} \in \mathbb{R}^{d'' \times ws \times d''}$. The weight matrix $\mathbf{W}$ contains d'' kernels, each of them has $ws \times d''$, convolving **ws** contiguous vectors. ws and d'' represent window-size and number of kernels respectively. Average pooling operation is added after final convolutional component, which can consolidate some unnecessary features and boost computational efficiency. sf is a scaling factor that weights the output with a constant factor. Hence, $\mathbf{X}''$ is trained on how much contribution it adds to the $\mathbf{X}$ using residual connection to improve the retrieval task. Final output $\mathbf{o} = [o_1, o_2, o_3, \cdots, o_{d''}] \in \mathbb{R}^{d''}$ is generated, which will be fed into the next component. Note that each of feature vector $\mathbf{o}$ is normalized to unit l_2 norm before passing to the next step.

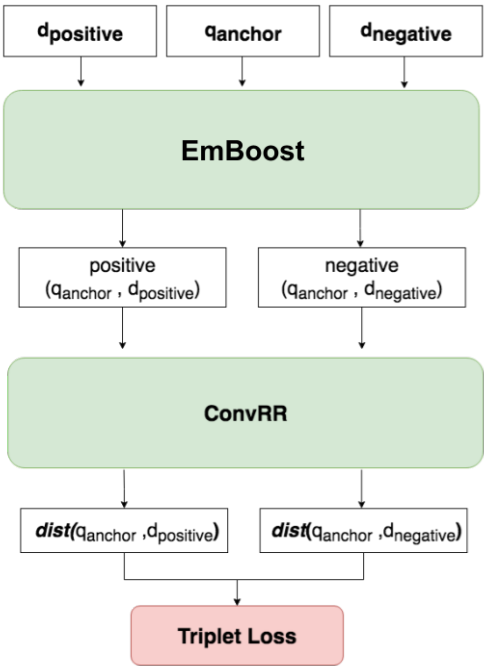

Figure 3: Overall flow diagram for the proposed approach.

3.4 Loss Function

In order to train the ConvRR network to perform well on retrieval task and generalize well on unseen data, we utilize the Siamese architecture with *triplet loss*—by (Hadsell et al., 2006), (Chopra et al., 2005)—during the training period as shown in Figure 3. With this setup, the network is encouraged to reduce distances between positive pairs so that they are lesser than distances between negative ones. A particular question $\boldsymbol{q}_{\text{anchor}}$ would be a question close in proximity to a document $\boldsymbol{d}_{\text{positive}}$ as the positive pair to the same question than to any document $\boldsymbol{d}_{\text{negative}}$ as they are positive pairs to other questions. The key point of the $\mathscr{L}_{\text{triplet}}$ is to build the correct triplet structure which should meet the condition of the following equation:

$$||\boldsymbol{q}_{\text{anchor}}, \boldsymbol{d}_{\text{positive}}|| + m < ||\boldsymbol{q}_{\text{anchor}}, \boldsymbol{d}_{\text{negative}}||$$

For each anchor, the positive $\boldsymbol{d}_{\text{positive}}$ is selected in such a way $\arg\max_{\boldsymbol{d}_{\text{positive}}} ||\boldsymbol{q}_{\text{anchor}}, \boldsymbol{d}_{\text{positive}}||$ and likewise the hardest negative $\boldsymbol{d}_{\text{negative}}$ in such a way that $\arg\min_{\boldsymbol{d}_{\text{negative}}} ||\boldsymbol{q}_{\text{anchor}}, \boldsymbol{d}_{\text{negative}}||$ to form a triplet. This triplet selection strategy is called *hard triplets mining*.

Let $T = (\boldsymbol{d}_{\text{positive}}, \boldsymbol{q}_{\text{anchor}}, \boldsymbol{d}_{\text{negative}})$ be a triplet input. Given T, the proposed approach computes the distances between the positive and negative pairs via a two-branch siamese subnet through the ensemble text embedding and ConvRR.

$$\mathscr{L}_{\text{triplet}} = [||\boldsymbol{q}_{\text{anchor}}, \boldsymbol{d}_{\text{positive}}|| - ||\boldsymbol{q}_{\text{anchor}}, \boldsymbol{d}_{\text{negative}}|| + m]^+ \tag{8}$$

where $m > 0$ is a scalar value, namely margin and $||.,.||$ represents the Euclidean distance between two vectors.

4 EXPERIMENTS

4.1 Datasets

In order to evaluate our proposed approach, we conducted extensive experiments on two large question-answering datasets, including SQuAD (Rajpurkar et al., 2016), and QUASAR (Dhingra et al., 2017).

4.1.1 SQuAD

The Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) is a large reading comprehension dataset that is built with $100,000+$ questions. Each of these questions are composed through crowdsourcing on a set of Wikipedia documents, where the answer to each question is a segment of text from the corresponding reading passage. In other words, the consolidation of retrieval and extraction tasks are aimed at measuring the success of the proposed systems.

4.1.2 QUASAR

The Question Answering by Search And Reading (QUASAR) is a large-scale dataset consisting of QUASAR-S and QUASAR-T. Each of these datasets is built to focus on evaluating systems devised to understand a natural language query, large corpus of text and to extract answer to the question from that corpus. Similar to SQuAD, the consolidation of retrieval and extraction tasks are aimed at measuring the success of the proposed systems. Specifically, QUASAR-S comprises $37,012$ fill-in-the-gaps questions that are collected from the popular website Stack Overflow, using entity tags. Since our research is not about addressing fill-in-the-gaps questions, we want to pay attention to the QUASAR-T dataset that fulfill the requirements of our focused retrieval task. The QUASAR-T dataset contains $43,012$ open-domain questions collected from various internet sources. The candidate documents for each question in this dataset are retrieved from an Apache Lucene based search engine built on the ClueWeb09 dataset (Callan et al., 2009).

The number of queries in each dataset, including their subsets, is listed in Table 1.

Table 1: Datasets Statistics: Number of queries in each train, validation, and test subsets.

DATASET	TRAIN	VALID.	TEST	TOTAL
SQUAD	87,599	10,570	HIDDEN	98,169+
QUASAR-T	37,012	3,000	3,000	43,012

4.2 Evaluation

The retrieval model aims to improve the *recall@k* score by selecting the correct pair among all candidates. Basically, *recall@k* would be defined as the number of correct documents as listed within top-*k* order out of all possible documents, (Manning et al., 2008). Additionally, embedding representations are visualized, using t-distributed stochastic neighbor embedding (van der Maaten and Hinton, 2008) in order to project the clustered distributions of the questions that are assigned to same documents.

4.3 Implementation

4.3.1 Input

Word embeddings were adopted, using the proposed ensemble text embedding, EmBoost. $f_{mixture}(\cdot,\cdot,\cdot)$ and $f_{ensemble}(\cdot)$ settings that represent the best configuration are shown in Table 2 and Table 3 respectively.

Table 2: $f_{mixture}(\cdot,\cdot,\cdot)$ configuration of EmBoost.

E	w_{idf}	m	f_{mix}	OUT
BERT	FALSE	$[\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0, .., 0]$	*concat.*	$\mathbf{x}^1$
ELMO	TRUE	[0, 0, 1]	*sum*	$\mathbf{x}^2$
FASTTEXT	TRUE	[1]	*sum*	$\mathbf{x}^3$

Table 3: $f_{ensemble}(\cdot)$ configuration of the multilevel abstract word embedding.

X'	u	$f_{ensemble}$
$\{\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3\}$	$[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$	*concat.*

The short form of this multilevel abstract word embedding is called as follows: $BERT \oplus ETwI \oplus FTwI$ where $(.)wI$ is denoting *"with IDF"* and $\oplus$ represents concatenation operation. The dimension of embedding is $4,372$.

4.3.2 ConvRR Cofiguration

ConvRR is trained, using ADAM optimizer (Kingma and Ba, 2014) with a learning rate of 10^{-3}. For the sake of equal comparison, we fixed the seed of randomization. We also observed that a weight decay of 10^{-3} tackles over-fitting. We choose windows-size $ws = 5$, number of kernel $d'' = 4,372$, and the scaling

factor $sf = 0.05$. We trained the network with 400 iterations with a batch size of $2,000$ using a triplet loss with a margin $m = 1$. Note that the best performance is achieved using a relative large batch size. All experiments are implemented with Tensorflow 1.8+ by (Abadi et al., 2015) on 2 $\times$ NVIDIA Tesla K80 GPUs.

5 RESULTS

We study different embedding models. We initialize text inputs of datasets, using different traditional embedding models. We first compared our model with the following baselines: TF-IDF, BERT, ELMo-AVG (averaging all layers of ELMo), GloVe, and fastText. Additionally, we also initialized text inputs, using the proposed EmBoost approach. To demonstrate the contribution of different components of EmBoost, we first configured it without using ensemble function, which includes ELMO-LSTM1 (first layer of ELMo), BERT w/IDF (BERT with IDF weight), ELMO-LSTM2 (second layer of ELMo), ELMO-TOKEN (token layer of ELMO), ELMO-TOKEN w/IDF (ELMo-TOKEN with IDF weight), FASTTEXT w/IDF (fastText with IDF weight). Then configured EmBoost with ensemble function of a concatenation of ELMo token layer with IDF weight and fastText with IDF weight (ETwI $\oplus$ FTwI), as well as a concatenation of BERT *(concatenation of last 4 layer representations)*, ELMo token layer with IDF weight, and fastText with IDF weight (BERT $\oplus$ ETwI $\oplus$ FTwI). Last but not least we also compared the performance gain using downstream models including a fully connected residual network (*FCRR*) and the convolutional residual network (*ConvRR*) respectively.

The *recall@k* results that calculated for SQuAD and QUASAR-T datasets are listed in Table 4 and Table 5. Our ConvRR model initialized with the proposed EmBoost approach outperforms all the baseline models on these datasets. More specifically the results show that the proposed EmBoost approach without ensemble function significantly improves the result of all baseline methods. The improvement is further increased when EmBoost uses ensemble function that combines embeddings trained using different models and corpora. The only exception is recall@1 for the result on QUASAR-T (in Table 5). We believe the reason is that QUASAR-T is a relatively small dataset, for which a single word embedding model trained using a very large corpus should be sufficient, and ensemble function is not necessary in this case. The best performance is achieved when the proposed EmBoost is applied with a downstream IR model, which is much better in comparison with the baseline word embedding models applied with the same downstream IR model.

Table 4: Experimental results on SQUAD. *recall@k* retrieved documents, using different models and the proposed approach.

EMBEDDING/MODEL	@1	@3	@5
BASE EMBEDDINGS			
TF-IDF	8.77	15.46	19.47
BERT	18.89	32.31	39.52
ELMO-AVG	21.24	36.24	43.88
GLOVE	30.84	47.14	54.01
FASTTEXT	42.23	59.86	67.12
EMBOOST (W/O ENSEMBLE)			
ELMO-LSTM1	19.65	34.34	42.52
BERT W/ IDF	21.81	36.35	43.56
ELMO-LSTM2	23.68	39.39	47.23
ELMO-TOKEN	41.62	57.79	64.36
ELMO-TOKEN W/ IDF	44.85	61.55	68.07
FASTTEXT W/ IDF	45.13	62.80	69.85
EMBOOST (W/ ENSEMBLE)			
ETwI $\oplus$ FTwI	46.33	63.13	69.70
BERT $\oplus$ ETwI $\oplus$ FTwI	48.49	64.96	71.05
BASE EMBEDDING + DOWNSTREAM MODELS			
FASTTEXT + FCRR	45.7	63.15	70.02
FASTTEXT + CONVRR	47.14	64.16	70.87
EMBOOST + DOWNS. MODELS			
BERT $\oplus$ ETwI $\oplus$ FTwI + FCRR	50.64	66.16	73.44
BERT $\oplus$ ETwI $\oplus$ FTwI + CONVRR	**52.32**	**68.26**	**75.68**

The t-SNE visualization of question embeddings that are derived, using different embedding models, including BERT, ELMo-TOKEN layer, fastText, multilevel abstract word embedding with a concatenation of BERT *(concatenation of last 4 layer representations)*, ELMo-TOKEN layer with IDF weight, and fastText with IDF weight, and ConvRR are shown in Figure 4. Note that those questions match the particular 4 (labeled as 57, 253, 531, 984) sampled contexts/documents that are extracted from SQuAD validation dataset. The visualization shows that the proposed EmBoost significantly improves the clustering of the questions and corresponding contexts/documents. The result is further improved by using ConvRR, the proposed retrieval model.

6 CONCLUSION

We developed a new multilevel abstract word embedding approach called EmBoost, which harnesses the power of individual strength of diverse word embedding methods. The performance of the proposed approach is further improved by using a convolutional residual retrieval model optimized using a triplet loss

Table 5: Experimental results on QUASAR-T. *recall@k* retrieved documents, using different models and the proposed approach.

EMBEDDING/MODEL	@1	@3	@5
BASE EMBEDDINGS			
TF-IDF	13.86	20.2	23.13
BERT	25.5	34.2	37.86
ELMO-AVG	27.93	37.86	42.33
GLOVE	32.63	40.73	44.03
FASTTEXT	46.13	56.00	59.46
EMBOOST (W/O ENSEMBLE)			
ELMO-LSTM1	24.6	33.01	36.9
ELMO-LSTM2	27.03	36.33	40.56
BERT W/ IDF	27.33	38.43	40.11
ELMO-TOKEN	44.46	54.86	59.36
ELMO-TOKEN W/ IDF	48.86	60.56	65.03
FASTTEXT W/ IDF	49.66	58.70	61.96
EMBOOST (W/ ENSEMBLE)			
ETWI ⊕ FTWI	48.78	60.05	64.10
BERT ⊕ ETWI ⊕ FTWI	49.46	60.93	65.66
BASE EMBEDDING + DOWNSTREAM MODELS			
FASTTEXT + FCRR	47.11	58.25	62.12
FASTTEXT + CONVRR	48.17	59.06	63.07
EMBOOST + DOWNS. MODELS			
BERT ⊕ ETWI ⊕ FTWI + FCRR	49.55	61.58	64.53
BERT ⊕ ETWI ⊕ FTWI + CONVRR	**50.67**	**63.09**	**67.38**

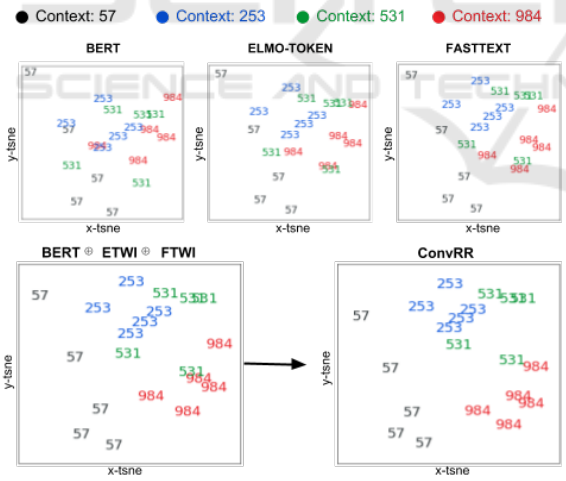

Figure 4: t-SNE map visualizations of various embedding models for all question representations of 4 (57, 253, 531, 984) sampled contexts/documents that are extracted from SQuAD validation dataset.

function for the task of document retrieval, which is a crucial step for many Natural Language Processing and information retrieval tasks. We further evaluate the proposed method for document retrieval from an unstructured knowledge base. The empirical study using large datasets including SQuAD and QUASAR benchmark datasets shows a significant performance gain in terms of the recall. In the future, we plan to apply the proposed framework for other information retrieval and ranking tasks. We also want to improve the performance of the retrieval task by applying and developing new loss functions and retrieval models.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Boom, C. D., Canneyt, S. V., Bohez, S., Demeester, T., and Dhoedt, B. (2015). Learning semantic similarity for very short texts. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1229–1234.

Cakaloglu, T., Szegedy, C., and Xu, X. (2018). Text embeddings for retrieval from a large knowledge base. *arXiv preprint arXiv:1810.10176*.

Cakaloglu, T. and Xu, X. (2019). MRNN: A multi-resolution neural network with duplex attention for document retrieval in the context of question answering. *CoRR*, abs/1911.00964.

Callan, J., Hoy, M., Yoo, C., and Zhao, L. (2009). Clueweb09 data set.

Cao, Q., Ying, Y., and Li, P. (2013). Similarity metric learning for face recognition. In *2013 IEEE International Conference on Computer Vision*, pages 2408–2415.

Chechik, G., Sharma, V., Shalit, U., and Bengio, S. (2010). Large scale online learning of image similarity through ranking. *J. Mach. Learn. Res.*, 11:1109–1135.

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. volume 1, pages 539–546 vol. 1.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Dhingra, B., Mazaitis, K., and Cohen, W. W. (2017). Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. CVPR '06, pages 1735–1742.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

Lowe, D. G. (1995). Similarity metric learning for a variable-kernel classifier. *Neural Computation*, 7(1):72–85.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2018). Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Perone, C. S., Silveira, R., and Paula, T. S. (2018). Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100, 000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks.

Salton, G. and McGill, M. J. (1986). Introduction to modern information retrieval.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2020). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. volume 00, pages 815–823.

Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. NIPS'16, pages 1857–1865.

Sun, Y., Chen, Y., Wang, X., and Tang, X. (2014). Deep learning face representation by joint identification-verification. pages 1988–1996.

Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. CVPR '14, pages 1701–1708.

van der Maaten, L. and Hinton, G. E. (2008). Visualizing data using t-sne.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., and Wu, Y. (2014). Learning fine-grained image similarity with deep ranking. pages 1386–1393.

Wang, J., Zhou, F., Wen, S., Liu, X., and Lin, Y. (2017). Deep metric learning with angular loss.

Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. 10:207–244.

Xing, E. P., Ng, A. Y., Jordan, M. I., and Russell, S. (2002). Distance metric learning, with application to clustering with side-information. NIPS'02, pages 521–528.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. (2020). Xlnet: Generalized autoregressive pretraining for language understanding.

Yu, A. W., Dohan, D., Luong, M.-T., Zhao, R., Chen, K., Norouzi, M., and Le, Q. V. (2018). Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.