Journal of Data Intelligence, Vol. 3, No. 1 (2021) 131–148

A COMPARATIVE STUDY USING DIFFERENT QUESTION CONTEXT INFORMATION ON PAIRWISE LEARNING-TO-RANK CQA TRANSFORMER MODELS IN THE HOME IMPROVEMENT DOMAIN

MACEDO MAIA
University of Passau
Passau, Germany
sousam02@gw.uni-passau.de

MARKUS ENDRES
University of Passau
Passau, Germany
Markus.Endres@uni-passau.de

To find answers for subjective questions about many topics through Q&A forums, questioners and answerers can cooperatively help themselves by sharing their doubts or answers based on their background and life experiences. These experiences can help machines redirect questioners to find better answers based on community question-answering models. This work proposes a comparative analysis of the pairwise community answer retrieval models in the home improvement domain considering different kinds of user question context information. Community Question-Answering (CQA) models must rank candidate answers in decreasing order of relevance for a user question. Our contribution consists of transformer-based language models using different kinds of user information to accurate the model generalisation. To train our model, we propose a proper CQA dataset in the home improvement domain that consists of information extracted from community forums, including question context information. We evaluate our approach by comparing the performance of each baseline model based on rank-aware evaluation measures.

Keywords: Information Retrieval, Community Question Answering, and Neural Networks.

1. Introduction

Web platforms allow web users to share opinions without disclosing too much identity. It encourages users to post genuine comments about something on web forums [17]. Some situations in the real world make some web users access the web to ask about daily problems. For instance, users who want to renovate their houses can ask for tips or opinions about improvements (e.g., buying new appropriate furniture or repairing some broken installation) in some specialized community forums.

The complexity of answering user questions is in the fact of lack of further explicit information to help models to find the most relevant answer. We show in Figure 1 an example of a subjective question and two user comments as accepted answers obtained from a community question-answering forum.[a] This example shows a web user asking some tips to measure the height of a tree. This questioner aims to install an antenna for internet service that needs

[a]https://diy.stackexchange.com/questions/7100/is-there-an-easy-way-to-measure-the-height-of-a-tree

to clear some tree because it requires a clear line of sight to work. The expected answers for this question are distinct because there are many ways that we must consider to measure the height of something. Answerer 1 suggests to find an approximated tree size by using some steps considering a pencil, moving some meters away from the tree, extend your arm, and hold the pencil so that you can measure the height of the tree on the pencil with your thumb. Meanwhile, Answerer 2 suggests to find the approximated tree height by using a formula based on the questioner's shadow, questioner's and the tree's shadow measures. We conclude that subjective questions have different answers based on the experience of different users. A possible solution to match question and correct answer is to find an association between "ways to measure" words and often terms that appear in texts that describe approach to measure something. A way to help to generalise pairwise CQA model inferences is to use further explicit information (e.g., tags) provided by users.

Question: *Is there an easy way to measure the height of a tree?*
Questioner's nickname: ***JohnFx***

Answer 1: *Take a pencil, move some meters away from the tree. Outstrech your arm and hold the pencil so that you can measure the height of the tree on the pencil with your thumb. Then turn the pencil at the bottom of the tree by 90 degrees. Note where the distance measured by thumb hits the earth and measure the way from this point to the tree. This is the height of the tree.*
Answerer's nickname 1: ***bennymo***

Answer 2: *Use shadows... 1-Measure your shadow; 2-Measure yourself; 3-Measure the tree's shadow. Calculate (tree's shadow * your height) / your shadow = Tree Height. You'll have to do this on a sunny day (you might also need an assistant), and the ground will have to be relatively flat (a slope will throw off the measurement).*
Answerer's nickname 2: ***Tester101***

Fig. 1. Examples of questions and their respective answers.

To provide further information for community questions, users can also post the question context to help users deeply understand which kind of answer the questioner expects and improve the question and answer matching or answer retrieval ranking models. In Figure 2 we show two different kinds of question context: question description and user tags. *Question description* is a full text that details the meaning of the question. *User tags* describe the more relevant keywords related to the question. These two kinds of information describe the context for a question differently. In Figure 2 we show the question description and the user tag for the question presented in Figure 1. In this question description, the questioner describes why it is necessary to measure the height of a tree. In this example, we only have a user tag. However, many questions have multiple different tags.

This work proposes a comparative analysis on community answer retrieval based on transformer learning-to-rank (LeToR) model in the home improvement domain. We propose scenarios based on question-based information (e.g., question, question description, and user tags) and QA pairwise sequence embedding matching.

As a first contribution, we propose a tag-based transformer model that improves the CQA learning-to-rank model generalisation. For that, we also propose an approach to encode

Question: *Is there an easy way to measure the height of a tree?*
Questioner's nickname: ***JohnFx***

Question Description: *I'm looking to get an antenna installed for Internet service that will need to clear some trees on my neighbor's property because it requires a clear line of sight to work. I really don't trust my estimating skills enough to plunk down money on a utility pole that might be too short or too tall. And climbing the tree with a tape measure (in my neighbors yard) is a bit intrusive and dangerous. Is there a clever way that I can get a reasonably close (within 5' or so) estimate of the height of the tree other than eyeballing it? I thought about putting together a bunch of 10' runs of PVC pipe and holding it up against the tree, but that is going to get pretty unwieldy by about 30' and the trees are at least that tall. My other thought is to break out my old Trig textbooks and use the angle/distance to tree, but it seems like it would be pretty hard to judge the angle correctly, maybe with a laser pointer or something? Any other ideas?*

User Tags: ***measuring***

Fig. 2. Question context information example.

multiple tags as a new sentence. As a second contribution, we build and validate a CQA dataset based on users questions and their respective candidate answers to test our approach. We also include a scenario where question context is part of question input. Finally, as a third contribution, we compare our approach with different state-of-the-art baselines in pairwise matching and pairwise learning-to-rank models using the rank-aware evaluation measures. The experiment aims to show the impact of each different question context information in each scenario.

The rest of the paper is organized as follows. Section 2 discusses related work. In Section 3 we present background on transformer models. Section 4 contains our approach to encode questions and answer pairs and include question context information. Section 5 describes the guidelines for building and validating our home improvement domain dataset. Section 6 presents our experiments and compares the results for different scenarios. We conclude in Section 7.

2. Related Work

Some CQA challenge series inspire our task [21, 20, 16] that aim to promote related research on that area. They provided datasets, annotated data, and developed robust evaluation procedures to establish a common ground for comparing and evaluating different CQA approaches. Despite these CQA challenge series contains user questions, they have no extra explicit information like user tags. This work is an extension of the work described in [15], where we consider question description as a new question context information to our comparative analysis. The plan is to evaluate whether the capacity of transformer model baselines improves the performance by including these kinds of new information. In order to explain how transformer models work, we first must understand how attention mechanisms work. In Section 3, we have a background about attention mechanisms and transformer architecture

applied in different transformer model variations.

Advances in word embedding models have improved neural network for generalising pairwise models. Word embedding refers to a group of Machine Learning algorithms that learn contextual high-dimensional dense word vector representations (e.g., Glove [22], Word2Vec [18], and ELMo [23]). There are some models based on Long Short-Term Memory (LSTM) [10] and the Gated Recurrent Unit (GRU) [4] to learn sequential information on words. Bilateral Multi-Perspective Matching Model (BiMPM) [27], for instance, proposes a model for text similarity. Given two sentences, P and Q, it matches the two encoded sentences in two directions, P against Q and Q against P. In each matching direction, each time step of one sentence is matched against all the other sentences' timesteps from multiple perspectives. Then, another BiLSTM layer is utilized to aggregate the matching results into a fixed-length matching vector. Finally, based on the matching vector, a decision is made through a fully connected layer. Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks (MPCNN) [8] proposes a model for comparing sentences that use diverse perspectives. Firstly, MPCNN models each sentence using a Convolutional Neural Network (CNN). It extracts features at multiple levels of granularity and uses multiple types of pooling. After, it compares the sentence representations at several granularities using multiple similarity metrics. We consider these last two approaches in our comparative experiments.

Unlike Recurrent Neural Network (RNN) and its extensions (e.g., LSTM and GRU) that process each sequence element, in turn, transformer-based language models [26] process all parts concurrently, forming direct connections between individual ones through attention mechanisms.

Bidirectional Encoder Representations from Transformers (BERT) [6] is the main transformer-based language model. It consists of pretraining deep bidirectional representations of unlabeled text by jointly conditioning both left and right context in all layers. Initially, that model outperforms other neural network approaches based on RNN, CNN and attentive mechanisms for eleven different NLP tasks. DistilBERT [25] is based on the BERT architecture with some optimisation procedures to improve the inference speed by removing some parameters to reduce the training speed and train a transformer model by considering a lower amount of computational resources. RoBERTa [12] also proposes a replication study of BERT that measures the impact of many key hyperparameters and training data size. The conclusion is that BERT was significantly undertrained, and must match or exceed the performance of some models in some tasks. RoBERTa specify some modifications in BERT pretraining step to get a better accuracy as well as decrease the time for pretraining a new model. The best model achieves state-of-the-art results on three Natural Language Processing (NLP) tasks.

XLNet [28] is similar to previously described transformer models, where tokens in XLNet are also predicted but in a random order. XLNet is based on two features: (1) enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order, and (2) overcomes the limitations of BERT thanks to its autoregressive formulation. As BERT, XLNet also model bidirectional contextual information, denoising autoencoding based pretraining. However, BERT neglects dependency between the masked positions and suffers from a pretrain-finetune discrepancy because it relying on corrupting the input with masks. XLNet also integrates ideas from Transformer-XL [5], the state-of-the-art autoregressive model, into pretraining to overcome this dependence. With the capability of

modeling bidirectional contexts, denoising autoencoding based pretraining like BERT achieves better performance than pretraining approaches based on autoregressive language modeling. However, relying on corrupting the input with masks, BERT neglects dependency between the masked positions and suffers from a pretrain finetune discrepancy.

3. Background

Although the RNN model and its variations (e.g., LSTM and GRU) provide accurate performance in some text classification tasks, they need to use special procedures like attention mechanisms to detect the most relevant parts (for prediction purposes) in texts. In order to address this issue, transformer-based models consist of multiple layers where each one contains multiple attention heads. To understand transformer models and Natural Language Processing (NLP) tasks like CQA, we must explore the background of their related topics. Our CQA baselines are based on attention-based neural networks. This section focuses on describing the background on attention mechanisms in NLP and the transformer architecture used as the base to understand how attention models work in our CQA baselines.

3.1. *Attention Mechanisms and NLP*

The first proposed attention mechanism approach was used in visual imaging in the 1990s [11]. However, attention became a hot topic after Google Mind studies, resulting in the work described in [19]. Attention mechanisms also became popular in applications about Neural Networks (NN) in other fields like Information Retrieval (IR) and Question-answering (Q&A) approaches. The NN model can align different modalities during the training procedure for different applications (e.g., aligning questions and answers to find the best QA pairwise matching). The attention mechanism is a part of a neural architecture that enables dynamically highlighting relevant features of the input data, which, in NLP, is typically a sequence of textual elements. It can be applied directly to the raw input or its higher-level representation. The core idea behind attention is to compute a weight distribution on the input sequence, assigning higher values to more relevant elements. The rapid advancement in modelling attention in Neural Networks on textual information is visible because these models are now the state-of-the-art for multiple tasks in NLP [7, 3].

The first successful applications of attentional mechanisms to align pairwise-matching texts task were used to improve Neural Machine Translation (NMT) by selectively focusing on parts of the source sentence during translation [14, 2]. The approach in [2] introduces an extension to the encoder-decoder model, which learns to align and translate jointly. For that, this work proposes a new architecture that consists of a bidirectional RNN as an encoder and a decoder that emulates searching for a set of positions in a source sentence during decoding a translation where the most relevant information receive a higher weight (Figure 3). The model predicts a target word based on two information: (1) the context vectors connected with the source positions and (2) the previously generated target words.

To annotate each word in a sentence, the model uses the bidirectional RNN to encode each word considering the word sequence information. For a word w_j, its annotation h_j (hidden state of the $j-th$ word) contains the summaries of both the preceding and the following words. The context vector c_i is computed as a weighted sum of these annotations h_i.

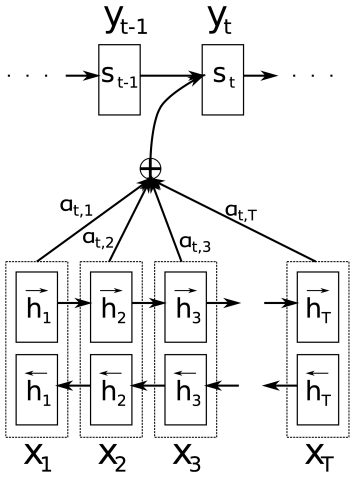

Fig. 3. An illustrative example of the attentive model architecture proposed in [2].

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j, \tag{1}$$

The attention weight α_{ij} of each annotation h_j is computed by:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T} exp(e_{ik})}, \tag{2}$$

$$e_{ij} = a(s_{i1}, h_j) \tag{3}$$

where a is an alignment model which scores the matching relevance between the inputs around position j and the output at position i and exp is an exponential function.

The target word y_t generation can be defined using conditional probability as:

$$p(y_i/y_1, \ldots, y_{i1}, X) = g(y_{i1}, s_i, c_i), \tag{4}$$

where g is a nonlinear, potentially multi-layered function that outputs the probability of y_i, and s_i is an RNN hidden state of the decoder for time i, computed by:

$$s_i = f(s_{i1}, y_i, c_i) \tag{5}$$

The probability α_{ij}, or its associated energy e_{ij}, reflects the importance of the annotation h_j with respect to the previous hidden state s_{i1} in deciding the next state s_i and generating y_i. Intuitively, this implements a mechanism of attention in the decoder.

NMT model has a sentence from a source language as an input and generate the corresponding sentence in another target language. The attention model helps the decoder finds the source input tokens that are relevant while generating the next token in the target sentence. For CQA, the application is the same. In that case, a use question is our source, and an answer is a target.

The approach described in [14] proposes some attention-based models separated into two different categories: *global* and *local*. These classes differ in whether the attention weight is placed on all source positions or only a few source positions. In Figure 4 we show the global and local attention model architectures. Common to these two types of models is that at each time step t in the decoding phase, both approaches first take as input the hidden state h_t at the top layer of a stacking LSTM. The goal is to derive a context vector c_t that captures relevant source-side information to help predict the current target word y_t. While these models differ in how the context vector c_t is derived, they share subsequent steps. In global attention mechanisms, at each time step t, the model infers a variable-length alignment weight vector a_t based on the current target state h_t and all source states $\tilde{h}_s$. A global context vector c_t is then computed as the weighted average, according to a_t. The model first predicts a single aligned position p_t for the current target word in local attention mechanisms. A window centred around the source position p_t is used to compute a context vector c_t (a weighted average of the source hidden states in the window). The weights a_t are inferred from the current target state h_t and that source states $\tilde{h}_s$ in the window, overall the source state. This attention is also used for machine translation task by considering the and we can implement this attentive approach to run in CQA in the same sense.

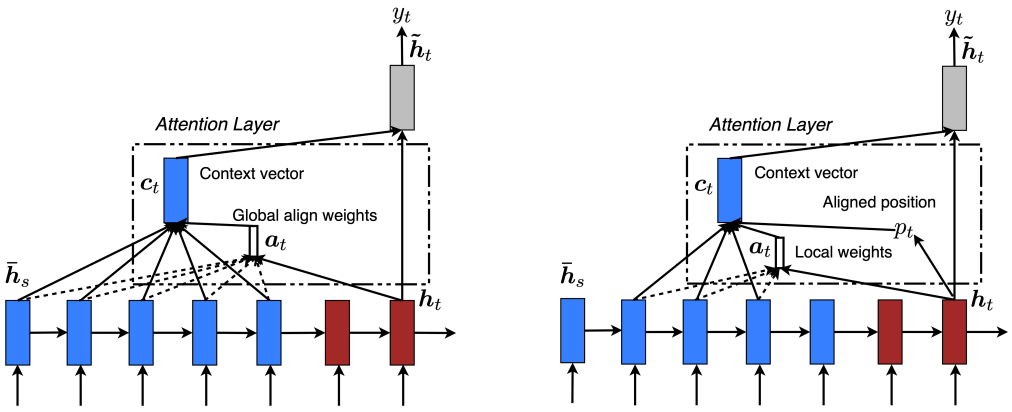

Fig. 4. Global and local attention model architectures presented in [14].

Attention-based models are used across a broad spectrum of problem domains. Due to their effectiveness, such models are top-rated in the language and vision domains, a global approach in which all source words are attended and a local one whereby only a subset of source words are considered at a time. The local and global attention mechanisms resemble the model of [2] but are simpler architecturally.

3.2. *Transformer Model Architecture*

In this section, we summarize the description of the transformer model architecture described in [26]. In transformer architecture, the encoder maps the source input sequence of symbol representations to a sequence of continuous representations. Given this continuos represen-

tations, the decoder then generates an target sequence of symbols one element at a time. At each step the model is auto-regressive [10], consuming the previously generated symbols as additional input when generating the next. [26]

In Figure 5 we find the transformer model architecture using stacked self-attention, point-wise, and fully connected layers for the encoder (left-hand side) and decoder (right-hand side). Initially, the transformer encoder is defined as a stacked similar layer (two sub-layers compound each). These sublayers are a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. A residual connection described in [9] and a layer normalization (LayerNorm) described in [1] are implemented between these sublayers. Similar to encoders, transformer decoders also are defined as a stacked similar layer with two sub-layers in each one. However, the decoder adds a extra sub-layer that process multi-head attention over the output of the stack encoder layer. Moreover, the decoder also applies residual connections and layer normalization in the same way they are applied in the encoder. The Self-attention sub-layer in the decoder stack was modified to prevent positions from attending to subsequent positions. This change, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.

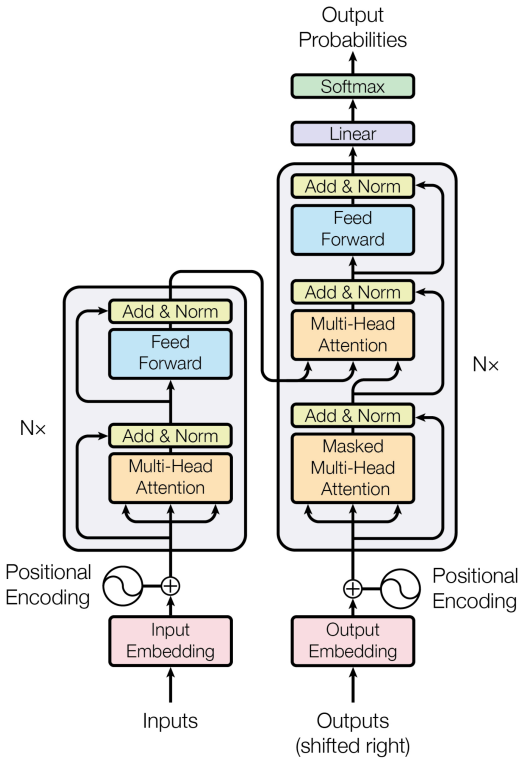

Fig. 5. Transformer Model Architecture (taken from [26]).

Transformer models are based on learned embeddings to convert input and output tokens into vectors. This model also computes the learned linear transformation and softmax function to transform the decoder output into information to predict the next-token probabilities. The same weight matrix are used by input, and output embedding layers and the pre-softmax linear transformation is based on [24].

3.3. *Attention in Transformers*

In transformer models, the attention mechanism functions are a mapping between a query and a set of key-value pairs to an output, where different vectors represent them. The output vector is represented as a weighted sum of the values. The weight in this sum is performed by a compatibility function of a query with its related key. In Figure 6 we show the self-attention models called Scaled Dot-Product Attention and Multiheads Attention, which are proposed as part of the transformer architecture.

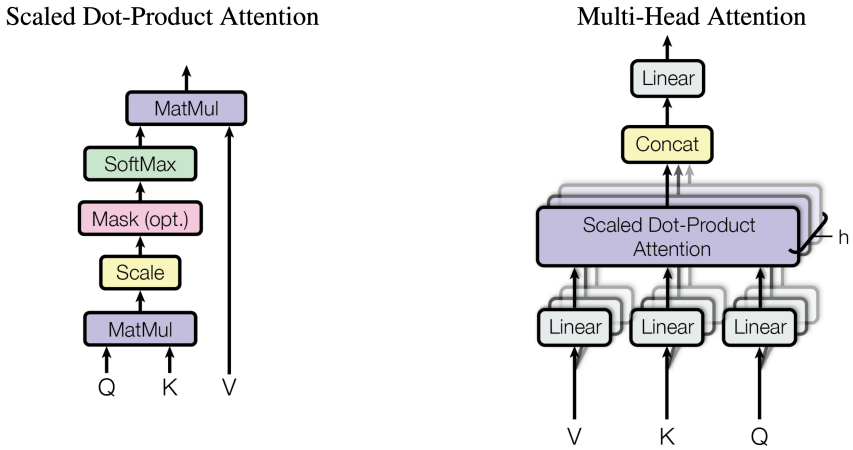

Fig. 6. Self-attention mechanisms used in transformer models described in [26]: On the left-hand side we show the Scaled Dot-Product Attention. On the right-hand side we show Multi-Head Attention consists of several attention layers running in parallel.

3.3.1. *Scaled Dot-Product Attention*

In this attention mechanism approach, the inputs are queries, keys and values where the dot products of the query are performed with all keys, divided by d_k. The softmax activation function is used to acquire the weights. The attention function is performed simultaneously on queries, inserted jointly into a matrix Q. The keys and values are also added into the matrices K and V. Equation 6 represents the attention function used to obtain the Matrix output.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6}$$

3.3.2. *Multiheads Attention*

Another way to apply attention is to linearly project the queries, keys and values h times with different, learned linear projections. In this case, the attention function is performed parallelly on these projected versions of queries, keys and values, resulting in output values. Then these are concatenated and once again projected, resulting in the final output.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. A single attention head inhibits this [26].

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{7}$$

$$head_i = Attention(QW_i^Q, QW_i^k, QW_i^V) \tag{8}$$

where W_i^Q, W_i^K, W_i^V, and W^O are learning parameters.

3.3.3. *Position-wise Feed-Forward Networks*

Each layer in the encoder and decoder has a fully connected Feed-Forward Network (FFN), which is applied to each position separately and identically [26].

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2, \tag{9}$$

where W_1, W_2, b_1, and b_2 are learning parameters.

4. Question Context Information on Transformer-based CQA Model

Given a question and a set of comments as candidate answers in natural language, a CQA LeToR model sorts comments as candidate answer list to the question in order of ascending relevance score. In Figure 7 we represent a generic QA pairwise LeToR model architecture.

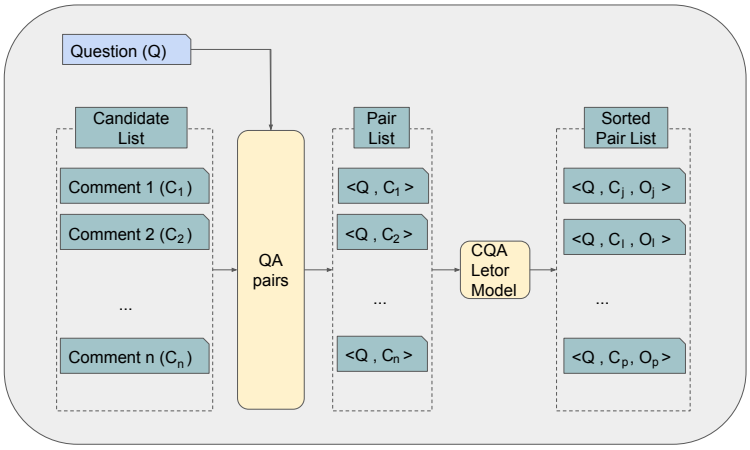

Fig. 7. CQA Learning-to-rank Pairwise Model

For a question Q and a set of n candidate answers $[C_1, C_2, C_3, ..., C_n]$ we must build question and comment pairs $QCP = \{< Q, C_1 >, < Q, C_2 >, < Q, C_3 >, ..., < Q, C_n >\}$, where The model output is represented by an n-dimensional vector $O = \{< C_1, O_1 >, < C_2, O_2 >, ..., < C_n, O_n >\}$, where the pair $< C_j, O_j >$ represents the comment C_j and the relevance score O_j of $< Q, C_j >$.

Our CQA pairwise learning-to-rank transformer model is based on a question answering pairwise approach, including different question context information as further information of questions.

4.1. *Input and Tag Representation*

The input in our proposed model is a user question $Q = \{w_1, w_2, ..., w_i, ..., w_m\}$, a candidate answer based on comment $C = \{v_1, v_2, ..., v_i, ..., v_n\}$, where w_i is the i-th token in Q and v_j is the j-th token in C. In order to include tags information in our model, we also consider extra inputs: the set of alphabetically ordered user tags $T = \{t_1, t_2, ..., t_o\}$ and a question description $D = \{x_1, x_2, ..., x_i, ..., x_m\}$.

To represent multiple tags as new information, we define a new input sentence adding the coordinating conjunction term "and" to connect all tags in alphabetical order (see Figure 8). Conjunction term helps the model identify different tags and avoid confusing them with compound names (e.g., electrical panel). We represent multiple tags in alphabetical order because it helps the model know which tags must appear before or after others. Question Q and tag-based sentence T are concatenated into a new input $\hat{Q}$ where Q comes before T (see Equation 10).

$$\hat{Q} \leftarrow [Q, T] \tag{10}$$

To represent question description as new information, we define a new input sentence by concatenating question Q and question description D (see Equation 11).

$$\hat{Q} \leftarrow [Q, D] \tag{11}$$

In our experiments, we considered tags and question description in different scenarios.

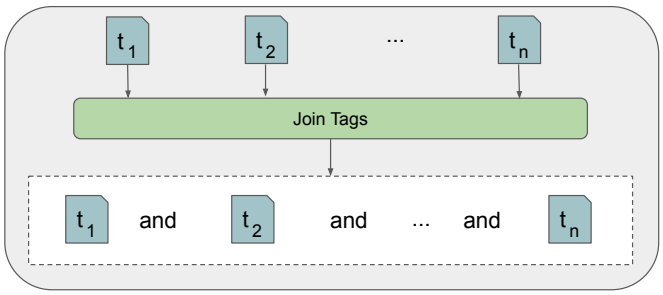

Fig. 8: Procedure to join tags and create a new sentence.

4.2. *CQA Pair Matching Model*

We use transformer-based language models to represent each QA pairwise representation using sequential contextual information among user question, tag-based sentence, and candidate answers.

For Information Retrieval (IR) tasks like CQA, we use pre-trained transformer models and run the fine-tuning step to fit the model parameters to our data. Firstly, we convert raw input question and answer pair into a proper unified input U. Before using a transformer tokeniser, we must indicate where the input starts using a special token "$< CLT >$". We also need to inform the model where the first sentence ends and where the second sentence begins by using a special token "$< SEP >$". The tokens $< CLT >$ and $< SEP >$ are predefined for the transformer models as default. After defining U, we use the Transformer Tokeniser ($T\_Tokeniser$) (Equation 12) to separate it into tokens.

$$U = [< CLT >, \hat{Q}, < SEP >, C, < SEP >] \quad ; \quad \hat{U} = T\_Tokeniser(U), \tag{12}$$

When we tokenise the inputs using Transformer Tokeniser, each token receives an ID. Hence, when we want to use a pre-trained transformer model, we need to convert each token in $\hat{U}$ into its corresponding unique IDs. The "attention mask" tells the model which tokens should be attended to and which must not. $Tokens\_id$ and $attention\_mask$ feed the input into the transformer model (Equation 13).

$$tokens\_id, att\_mask = \hat{U}.convert\_tokens\_to\_ids(), \hat{U}.get\_att\_mask() \tag{13}$$

After defining the word encodings and the attention mask, we must pass these as input parameters to the transformer model (Equation 14). The transformer output is the addictive pairwise encoding by considering whether an answer comes after a question. In Section 3 we described information about transformer models. Finally, We feed the result to a *sigmoid* activation function in the output layer (Equation 14) to return the pairwise matching relevance score.

$$M = Transformer(tokens\_id, att\_mask) \quad ; \quad o_{rel} = \sigma(W_h M + b) \tag{14}$$

where $W_h \in \mathbb{R}^{|M|}$ and $b \in \mathbb{R}$ are learning parameters and σ represents a sigmoid function. The variable o_{rel} represents the relevance score of U where $\{o_{rel} \in \mathbb{R} \quad | \quad 0 \leq o_{rel} \leq 1\}$.

4.3. *Model Optimisation*

We use the samples in the training set described in Section 5 for training the model by minimising the binary cross-entropy loss:

$$Loss = -\frac{1}{|S|} \sum_{s=1}^{|S|} y_s \log \hat{y}_s + (1 - y_s) log(1 - \hat{y}_s), \tag{15}$$

where $|S|$ is the size of the training dataset. y_s and $\hat{y}_s$ are the true and the predicted CQA pairwise relevance, respectively. The model parameters are optimised through the AdamW optimizer [13].

4.4. *Candidate Answers Ranking*

Our pairwise LeToR model must rank all the QA pairs for a question Q in terms of relevance. Let L the list of all QA pairs for q, adding the all pair relevance scores. The result is shown in Equation 16:

$$L = [< q, c_1, o_1 >, < q, c_2, o_2 >, ..., < q, c_n, o_n >] \quad ; \quad \hat{L} = desc\_sort(L), \tag{16}$$

where $|L| = n$, where n is the number of candidate answers to q and o_j is the relevance score of the pair $< q, c_j >$. To ranking our QA pairs, we sort L in relevance score descending order through the function *desc_sort*, where the final output is a sorted list $\hat{L}$ and the most relevant pair is in the first position.

5. Dataset Building and Validation

In this section, we describe the procedure of building our CQA dataset and its validation.

5.1. *Subjective CQA*

We built our dataset composed of subjective texts harvested from a specialized Q&A forum called Home Improvement (HI)[b]. This website has 77.000 thousands of subscribed users with some background in this domains. They share questions and opinions about different subjects associated with home improvements. Some moderators review new postings, correct mistakes directly and suggest new changes for original authors. They use facts and references to support and help the user to fix some incoherence or misunderstanding information.

Postings in these websites also contain some explicit information provided by users. We use the number of upvotes as information to validate our gold standard dataset. Upvotes represents the number of positive votes that the message receive by other users. we build our dataset considering the two information. We scrapped the question comments from both home improvements forums in a dataset.

We describe quantitative information about questions and their associated tags in Table 1. Following the information in these tables, we state that 11935 questions about home improvement topics where each question is associated with one tag, at least. There are questions associated with five different user tags.

Table 1. Distribution of number of tags by number of questions in either training, dev, and test sets in home improvement dataset.

Datasets	**#1_tag**	**#2_tags**	**#3_tags**	**#4_tags**	**#5_tags**	**#Questions**
Training	2308	2681	1673	627	187	7476
Validation	250	546	563	248	125	1732
Test	196	757	961	563	250	2727
Total	2754	3984	3197	1438	562	11935

Questions are associated with user tags. We show in Table 2 the most frequent user tags in our dataset. Precisely, we have 771 different home improvement tags in our dataset.

[b] https://dyi.stackexchange.com

Table 2. Tag examples and their frequency for our Home Improvements dataset.

tag	frequency
electrical	2941
wiring	1136
plumbing	827
lighting	423
drywall	392
bathroom	358

5.2. *Gold Standard Definition*

We distributed all the questions into training, dev and test datasets in question amount. Initially, we distributed all the questions using an overall distribution of about 75/25% for the training and test dataset. Afterwards, we split the original training set into training and development sets with a proportion of 80/20% approximately. In Table 3 we show the distribution of the number of questions, number of accepted answers, number of non-related answers, and QA pairs.

Table 3. Distribution of the number of questions by labels in either training, dev and test datasets for home improvements.

Datasets	#Question	#Acc_Answers	#NonRelev_Answers	#QA_pairs
Training	7476	10451	29542	39993
Validation	1732	2408	6879	9287
Test	2727	3878	10856	14734
Total	11935	16737	47277	64014

6. Evaluation

We evaluated the ranking performance of our approach with different state-of-the-art learning-to-rank models on our gold standard datasets described in Section 5.

6.1. *Experimental Setup*

For Recurrent Neural Networks, we performed our experiments with the 300-dimension word vectors by using the pre-trained GloVE [22] word embedding model generated from a 6-billion-token corpus[c](extracted from Wikipedia 2014 + Gigaword 5). Regarding transformer-based language models baselines, we used different configurations for each one. In Table 4 we describe the pretraining model configuration for different transformer-based algorithms. The RNN- and CNN-based models' parameters were optimized through the Adam optimizer, and transformer-based language models optimized through the AdamW optimizer [13].

We ran our experiments in a dedicated server with Intel(R) Xeon(R) CPU E5-2643 v3 with 24 cores, 128 GB DIMM DDR4 Memory, and an NVIDIA Tesla K40c GPU device.

[c]http://nlp.stanford.edu/data/glove.6B.zip

Table 4. Overview of transformer-based language models configuration.

Model	pre-trained	#layer	#hidden	#param	#token
BERT	bert-base-uncased	12	768	110M	512
RoBERTa	roberta-base	12	768	125M	512
DistilBERT	distilbert-base-uncased	6	768	66M	512
XLNet	xlnet-base-cased	12	768	110M	512

6.2. *Rank-aware Evaluation Metrics*

We considered two rank-aware evaluation measures in our experiments. Mean Reciprocal Rank (MRR) is the average of the reciprocal ranks of the first relevant answer among candidate answers for a set of community queries. Mean Average Precision (MAP) is the mean of average precision across multiple community questions. For a single question, Average Precision (AP) is the average of the precision value obtained for the set of top-k candidate answers existing after each relevant answer. MRR evaluates the performance of a ranking-based model to find the first relevant answer. In contrast, MAP evaluates the capacity of the same model to predict the whole candidate answer list.

6.3. *Results*

In Table 5 we show the performance of different baselines considering different evaluation measures for the ranking task. Among RNN-based neural network models, BiPMP has the best performance in terms of ranking measures. BiPMP outperformed MPCNN and the simple Siamese BiLSTM for both evaluation measures. BiPMP also outperformed MPCNN with 6.1% of the difference.

The previously discoursed approaches depend on the word embedding models to encode the model. Real-world texts are a problem because this kind of text contains some new words or misspelt words classified as an out-of-vocabulary words. These word groups have no distinct representation, and thus the model must lose relevant information. Unlike the previously described models, transformer-based language models performed better in all scenarios. XLNet outperforms all the state-of-the-art approaches considered in our analysis. However, the difference in performance for each transformer-based model is slight, as shown in Table 5 for XLNet, BERT and DistilBERT. XLNet obtained the best results in our ranking analysis among these baseline models.

6.4. *Question Context Influence on Transformer-based Pairwise CQA task*

In the same Table 5, we realised how transformer models generalise better when we include user tags or question descriptions as extra information on questions. For DistilBERT and BERT with user tags information, for instance, outperformed these models without this new information in around 2% of difference for both MRR and MAP measures. Although XLNet with multiple user tags also has improvements by generalising ranking models regarding the same model, the difference is smaller than the other two transformer models without this extra information. The explanation is in the way how attention mechanisms work for each model. DistillBert and BERT have the same parameters where the difference is because the first one is a distilled version of the second one (it must explain why BERT outperforms DistillBERT).

Unlike BERT and DistillBERT, XLNet outperforms the results of other approaches without using tags as extra information. However, Bert and DistilBERT obtained more advantages in extracting information on tags than XLNet. Unlike BERT, DistillBERT and XLNet, Roberta has no improvements when using tags as extra information. BERT outperformed the other approaches for transformer models using tags as extra information.

Regarding the model performance, when we include question description as extra information, we find out that it improved the performance of rank-based community answer retrieval considering transformer models for almost every transformer-based model. The reason is that the question description specifies better the question context where the questioner explains doubt in further detail. In that case, RoBERTa outperformed all the other baselines where it was the unique model setting that reached 90% for MRR measure. RoBERTa also had the most significant improvement concerning comparing the transformer models with and without question context as extra information. XLNET and BERT also had a relevant improvement in that sense.

Table 5. Experiment results for Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR).

	Home Improv.	
	MRR	**MAP**
MPCNN	0.636	0.617
BiPMP	0.701	0.676
DistilBERT	0.773	0.753
DistilBERT+ tags	**0.791**	**0.773**
DistilBERT+question description	0.788	0.769
BERT	0.799	0.778
BERT+ tags	0.821	0.804
BERT+question descriptiont	**0.868**	**0.848**
XLNet	0.813	0.795
XLNet+tags	0.815	0.798
XLNet+question description	**0.888**	**0.872**
RoBERTa	0.823	0.808
RoBERTa+ tags	0.779	0.756
RoBERTa+question description	**0.900**	**0.885**

7. Conclusion

We proposed a comparative analysis on community answer retrieval based on the transformer LeToR model in the home improvement domain. As a first contribution, we built a CQA dataset based on users questions and comment-based answers. We validated that dataset by considering explicit information and feedback from different users on the website. We also proposed a way to encode question and answer pairs considering question, answers, and question context information. Our experiments show that question context input helps increase the CQA generalisation performance on pairwise LeToR CQA tasks considering different transformer-based models. BERT showed better performance using the tag information if we compare the difference between the performance of each approach without tags and with these. Approaches using question description as an extra input information present the best

results where RoBERTa outperformed every baseline in this scenario.

We plan to analyse community questions from other domains and determine the effects of specific-domain multiple tag information in the model generalisation for future work. We also plan to explain the effects of tag information on the attention mechanism for transformer models.

Reference

1. Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.
2. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
3. Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. An attentive survey of attention models. *ACM Trans. Intell. Syst. Technol.*, 12(5), 2021.
4. Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.
5. Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.
6. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina 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*, pages 4171–4186, Minneapolis, Minnesota, 2019.
7. Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention in natural language processing. *IEEE Trans. Neural Networks Learn. Syst.*, 32(10):4291–4308, 2021.
8. Hua He, Kevin Gimpel, and Jimmy Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1576–1586, Lisbon, Portugal, September 2015. Association for Computational Linguistics (ACL).
9. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
10. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 1997.
11. L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.
12. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, 2019.
13. Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization. In *Procedings of Seventh International Conference on Learning Representations (ICLR 2019)*, 2019.
14. Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.
15. Macedo Maia, Siegfried Handschuh, and Markus Endres. A tag-based transformer community question answering learning-to-rank model in the home improvement domain. In Christine Strauss, Gabriele Kotsis, A Min Tjoa, and Ismail Khalil, editors, *Database and Expert Systems Applications - 32nd International Conference, DEXA 2021, Virtual Event, September 27-30, 2021, Proceedings, Part II*, volume 12924 of *Lecture Notes in Computer Science*, pages 127–138. Springer, 2021.

16. Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. Www'18 open challenge: Financial opinion mining and question answering. In Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis, editors, *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, pages 1941–1942. ACM, 2018.
17. Macedo Maia, Juliano E. Sales, André Freitas, Siegfried Handschuh, and Markus Endres. A comparative study of deep neural network models on multi-label text classification in finance. In *15th IEEE International Conference on Semantic Computing, ICSC 2021, Laguna Hills, CA, USA, (online) January 27 - 29, 2021*. IEEE Computer Society, 2021.
18. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013.
19. Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2204–2212, Cambridge, MA, USA, 2014. MIT Press.
20. Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. SemEval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 27–48, Vancouver, Canada, August 2017. Association for Computational Linguistics.
21. Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. SemEval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 525–545, San Diego, California, June 2016. Association for Computational Linguistics.
22. Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.
23. Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.
24. Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceesings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 157–163, 2017.
25. Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC$\hat{2}$ Workshop*, 2019.
26. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, 2017.
27. Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 4144–4150. AAAI Press, 2017.
28. Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763, 2019.