Preferred Answer Selection in Stack Overflow: Better Text Representations ... and Metadata, Metadata, Metadata

Xingyi Xu, Andrew Bennett, Doris Hoogeveen, Jey Han Lau, Timothy Baldwin
The University of Melbourne
{stevenxxiu,awbennett0,doris.hoogeveen,jeyhan.lau}@gmail.com, tb@ldwin.net

Abstract

Community question answering (cQA) forums provide a rich source of data for facilitating non-factoid question answering over many technical domains. Given this, there is considerable interest in answer retrieval from these kinds of forums. However this is a difficult task as the structure of these forums is very rich, and both metadata and text features are important for successful retrieval. While there has recently been a lot of work on solving this problem using deep learning models applied to question/answer text, this work has not looked at how to make use of the rich metadata available in cQA forums. We propose an attention-based model which achieves state-of-the-art results for text-based answer selection alone, and by making use of complementary metadata, achieves a substantially higher result over two reference datasets novel to this work.

1 Introduction

Community question answering ("cQA") forums such as Stack Overflow have become a staple source of information for technical searches on the web. However, often a given query will match against multiple questions each with multiple answers. This complicates technical information retrieval, as any kind of search or question-answering engine must decide how to rank these answers. Therefore, it would be beneficial to be able to automatically determine which questions in a cQA forum are most relevant to a given query question, and which answers to these questions best answer the query question.

One of the challenges in addressing this problem is that cQA threads tend to have a very rich and specific kind of structure and associated metadata. The basic structure of cQA threads is as follows: each thread has a unique question (usually editable by the posting user) and any number of answers to that question (each of which is usually editable by the posting user); comments can be posted by any user on any question or answer, e.g. to clarify details, challenge statements made in the post, or reflect the edit history of the post (on the part of the post author); and there is some mechanism for selecting the "preferred" answer, on the part of the user posting the original question, the forum community, or both. There is also often rich metadata associated with each question (e.g. number of views or community-assigned tags), each answer (e.g. creation and edit timestamps), along with every user who has participated in the thread — both explicit (e.g. badges or their reputation level) and implicit (e.g. activity data from other threads they have participated in, types of questions they have posted, or the types of answers they posted which were accepted).

Our research is aimed at improving the ability to automatically identify the best answer within a thread for a given question, as an initial step towards cross-thread answer ranking/selection. In this work we use Stack Overflow as our source of cQA threads. More concretely, given a Stack Overflow cQA thread with at least four answers, we attempt to automatically determine which of the answers was chosen by the user posting the original question as the "preferred answer".

A secondary goal of our research is learning how to leverage both the question/answer text in cQA threads, along with the associated metadata. We show how to create effective representations of both the thread text and the metadata, and we investigate the relative strength of each as well as their complementarity for preferred answer selection. By leveraging this metadata and using an attentional model for constructing question/answer pair representations, we are able to obtain greatly improved results over an existing state-of-the-art method for answer retrieval.

The contributions of our research are as follows:

- we develop two novel benchmark datasets for cQA answer ranking/selection;
- we adapt a deep learning method proposed for near-duplicate/paraphrase detection, and achieve state-of-the-art results for text-based answer selection; and
- we demonstrate that metadata is critical in identifying preferred answers, but at the same time text-based representations complement metadata to achieve the best overall results for the task.

The data and code used in this research will be made available on acceptance.

2 Related work

The work that is most closely related to ours is Bogdanova and Foster (2016) and Koreeda et al. (2017). In this first case, Le and Mikolov's paragraph2vec was used to convert question–answer pairs into fixed-size vectors in a word-embedding vector space, which were then fed into a simple feed-forward neural network. In the second case, a decompositional attentional model is applied to the SemEval question–comment re-ranking task, and achieved respectable results for text alone. We improve on the standalone results for these two methods through better training and hyperparameter optimisation. We additionally extend both methods by incorporating metadata features in the training of the neural model, instead of extracting neural features for use in a non-deep learning model, as is commonly done in re-ranking tasks (Koreeda et al., 2017).

In addition to this, there is a variety of other recent work on deep learning methods for answer ranking or best answer selection. For instance, Wang et al. (2010) used a network based on restricted Bolzmann machines (Hinton, 2002), using binary vectors of the most frequent words in the training data as input. This model was trained by trying to reconstruct question vectors from answer vectors, then at test time question vectors were compared against answer vectors to determine their relevance.

Elsewhere, Zhou et al. (2016) used Denoising Auto-Encoders (Vincent et al., 2008) to learn how to map both questions and answers to low-dimensional representations, which were then compared using cosine similarity. The resulting score was used as a feature in a learn-to-rank setup, together with a set of hand-crafted features including metadata, which did not have a positive effect on the results.

In another approach, Bao and Wu (2016) mapped questions and answers to multiple lower dimensional layers of variable size. They then used a 3-way tensor transformation to combine the layers and produce one output layer.

Nassif et al. (2016) used stacked bidirectional LSTMs with a multilayer perceptron on top, with the addition of a number of extra features including a small number of metadata features, to classify and re-rank answers. Although the model performed well, it was no better than a far simpler classification model using only features based on text (Belinkov et al., 2015).

Compared to these past deep learning approaches for answer retrieval, our work differs in that we include metadata features directly within our deep learning model. We include a large number of such features and show, contrary to the results of previous research, that they can greatly improve classification performance.

In addition to deep learning methods for answer retrieval, there is plenty of research on answer selection using more traditional methods. Much of this work involves using topic models to infer question and answer representations in topic space, and retrieving based on these representations (Vasiljevic et al., 2016; Zolaktaf et al., 2011; Chahuara et al., 2016). However, the general finding is that this kind of method is insufficient to capture the level of detail needed to determine if an answer is truly relevant (Vasiljevic et al., 2016). They therefore tend to rely on complementary approaches such as using translation-based language models (Xue et al., 2008), or using category information. Given this, we do not experiment with these kinds of approaches.

There is also some work on improving answer retrieval by directly modelling answer quality (Jeon et al., 2006; Omari et al., 2016; Zhang et al., 2014). User-level information has proven to be very useful for this (Agichtein et al., 2008; Burel et al., 2012; Shah, 2015), which helps motivate our use of metadata.

Finally, an alternative strategy for answer selection is analogical reasoning or collective classification, which has been investigated by Tu et al.

(2009), Wang et al. (2009) and Joty et al. (2016). In this kind of approach, questions and their answers are viewed as nodes in a graph connected by semantic links, which can be either positive or negative depending on the quality of the answer and its relevance to the question. However, we leave incorporating such graph-based approaches to future work.

3 Dataset

We developed two datasets based on Stack Overflow question–answer threads, along with a background corpus for pre-training models.[1] The evaluation datasets were created by sampling from threads with at least four answers, where one of those answers had been selected as "best" by the question asker.[2] The process for constructing our dataset was modelled on the 10,000 "how" question corpus (Jansen et al., 2014), similar to Bogdanova and Foster (2016).

The two evaluation datasets, which we denote as "SMALL" and "LARGE", contain 10K and 70K questions, respectively, each with a predefined 50/25/25 split between `train`, `val`, and `test` questions. On average, there are approximately six answers per question.

In addition to the sampled sub-sets, we also used the full Stack Overflow dump (containing a full month of questions and answers) for pre-training; we will refer to this dataset as "FULL". This full dataset consists of approximately 300K questions and 1M answers. In all cases, we tokenised the text using Stanford CoreNLP (Manning et al., 2014).

Stack Overflow contains rich metadata, including user-level information and question- and answer-specific data. We leverage this metadata in our model, as detailed in Section 4.2. Summary statistics of SMALL, LARGE and FULL are presented in Table 1.

In addition to the Stack Overflow dataset, we also experiment with an additional complementary dataset: the SEMEVAL 2017 Task 3A Question-Comment reranking dataset (Nakov et al., 2017).

[1]All the data was drawn from a dump dated 9/2009, which has a month of Stack Overflow question and answers.

[2]Note that in Stack Overflow, the community can separately vote for answers, with no guarantee that the top-voted answer is the preferred answer selected by the question asker. In this research — consistent with Bogdanova and Foster (2016) — we do not directly train on the vote data, but it could certainly be used to fully rank answers.

Model	SMALL	LARGE	FULL
Questions	10,000	70,000	314,731
Answers	64,671	457,634	1,059,253
Comments	70,878	493,020	1,289,176
Words	9,154,812	64,560,178	174,055,024
Vocab size	218,683	962,506	2,428,744

Table 1: Details of the three Stack Overflow datasets.

We include this dataset to establish the competitiveness of our proposed text processing networks (noting that the data contains very little metadata to be able to evaluate our metadata-based model). We used the 2016 test set as validation, the 2017 test set as test. Note that there are 3 classes in SEMEVAL: `Good`, `PotentiallyUseful`, and `Bad`, but we collapse `PotentiallyUseful` and `Bad` into a single class, following most competition entries.

4 Methodology

We treat the answer ranking problem as a classification problem, where given a question/answer pair, the model tries to predict how likely the answer is to be the preferred answer to the question. So for a given question, the answers are ranked by descending probability.

We explore three methods, which vary based on how they construct a question/answer pair embedding. Respectively these variations leverage: (1) only the question and answer text; (2) only the metadata about the question, answer and users; or (3) both text and metadata.

In all cases, given a vector embedding of a question/answer pair (based on a text embedding and/or metadata embedding), we feed the vector into a feed-forward network, H, which outputs the probability that the answer is the preferred answer to the given question. The network H consists of a series of dense layers with `relu` activations, and a final `softmax` layer. The model is trained using SGD with standard categorical cross-entropy loss, and implemented using TensorFlow.[3]

4.1 Text Only

We experiment with two methods for constructing our text embeddings: an attentional approach, and a benchmark approach using a simple paragraph vector representation.

[3]https://www.tensorflow.org/

4.1.1 Decompositional Attentional Model

Parikh et al. (2016) proposed a decompositional attentional model for identifying near-duplicate questions. It is based on a bag-of-words model, and has been shown to perform well over the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015; Tomar et al., 2017).

We adapt their architecture for our task, running it on question/answer pairs instead of entailment pairs. Note that, in our case, the best answer is in no way expected to be a near-duplicate of the question, and rather, the attention mechanism over word embeddings is used to bridge the "lexical gap" between questions and answers (Shtok et al., 2012), as well as to automatically determine the sorts of answer words that are likely to align with particular question words. Henceforth we refer to our adapted model as "`decatt`".

The model works as follows: first it attends words mutually between the question and answer pair. Then, for each word in the question (respectively answer), it computes a weighted sum of the word embeddings in the answer (respectively question) to generate a soft-alignment vector. The embedding and alignment vector of each word are then combined together (by concatenation and feed-forward neural network) to form a token-specific representation for each word. Finally, separate question/answer vectors are constructed by summing over their respective token representations, and these are concatenated to form the final question/answer pair vector.

Formally, let the input question and answer be $\mathbf{a} = (a_1, ..., a_{l_a})$ and $\mathbf{b} = (b_1, ..., b_{l_b})$ with lengths l_a and l_b, respectively. $a_i, b_j \in \mathbb{R}^d$ are word embeddings of dimensionality d. These embeddings are not updated during training, following Parikh et al. (2016).

We first align each question (answer) word with other answer (question) words. Let F be a feed-forward network with `relu` activations. We define the unnormalised attention weights as follows: $e_{i,j} := F(a_i)^\intercal F(b_j)$.

We then perform `softmax` over the attention weights and compute the weighted sum:

$$\beta_i := \sum_{j=1}^{l_b} \frac{\exp(e_{i,j})}{\sum_{k=1}^{l_b} \exp(e_{i,k})} b_j$$

$$\alpha_j := \sum_{i=1}^{l_a} \frac{\exp(e_{i,j})}{\sum_{k=1}^{l_a} \exp(e_{k,j})} a_i$$

Let G be a feed-forward network with `relu` activations. We define the representation for each word as follows:

$$\mathbf{v}_{1,i} := G([a_i; \beta_i]); \quad \mathbf{v}_{2,j} := G([b_j; \alpha_j])$$

for $i = 1, ..., l_a$, $j = 1, ..., l_b$, and where $[\cdot;\cdot]$ denotes vector concatenation. Lastly, we aggregate the vectors in the question and answer by summing them:

$$\mathbf{v}_1 = \sum_{i=1}^{l_a} \mathbf{v}_{1,i}; \quad \mathbf{v}_2 = \sum_{j=1}^{l_b} \mathbf{v}_{2,i}$$

Finally, we concatenate both vectors, $\mathbf{v}_{\text{text}} = [\mathbf{v}_1; \mathbf{v}_2]$. This text vector is used as the input in the classification network H.

4.1.2 Paragraph Vectors

Our second approach uses the method of Bogdanova and Foster (2016), who achieved state-of-the-art performance on the Yahoo! Answers corpus of Jansen et al. (2014). The method, which we will refer to as "`doc2vec`", works by independently constructing vector representations of both the question and answer texts, using paragraph vectors (Le and Mikolov, 2014; Lau and Baldwin, 2016) in the same vector space. The training is unsupervised, only requiring an unlabelled pre-training corpus to learn the vectors.

The `doc2vec` method is an extension of `word2vec` (Mikolov et al., 2013) for learning document embeddings. The document embeddings are generated along with word embeddings in the same vector space. `word2vec` learns word embeddings that can separate the words appearing in contexts of the target word from randomly sampled words, while `doc2vec` learns document embeddings that can separate the words appearing in the document from randomly sampled words.

Given the `doc2vec` question and answer vectors, we concatenate them to construct the text vector, $\mathbf{v}_{\text{text}}$, which is used as the input to H. Note that in this model $\mathbf{v}_{\text{text}}$ is kept fixed after pre-training (unlike in `decatt` where errors are propagated all the way back to the $\mathbf{v}_{\text{text}}$ vectors).

4.2 Metadata Only

In order to leverage the metadata in the Stack Overflow dataset, we extract a set of features to form a fixed-length vector as input to our model. Given the wide difference in scale of these features, all feature values are linearly scaled to the range $[0, 1]$. We denote this vector as $\mathbf{v}_{\text{meta}}$, and in the metadata-only case this is used as the input to the classification network H.

The raw metadata is as follows: firstly, for each question and answer we used the number of times the post had been viewed, the creation date of the post, the last activity date on the post, and a list of comments on the post, including the user ID for each comment. Secondly, for each question we used the top n tags for the question (based on the number of community votes), where n is a tunable hyperparameter. Finally, for each user we used the account creation date, number of up/down votes, reputation score, and list of badges obtained by the user.[4]

From these raw metadata fields we constructed sets of question-specific, answer-specific, and user-specific features, which are summarised in Table 2. All date features were converted to integers using seconds since Unix epoch, and all binary features were converted to zero or one. In addition, the tag-based features were converted to a probability distribution based on simple MLE.[5]

One concern with this model is that concatenating all features together could lead to feature groups with lots of features dominating groups with fewer features (for example the `BasicQ` and `BasicA` features could be overshadowed by the `QTags` and `UTags` features). In order to control for this, we only used the top n tags for the `QTags` and `UTags` feature groups.

A further possible concern is that, in a real-world scenario, not all of this metadata would be available at classification time (e.g. some of it is generated quite a bit after the questions and answers are posted). In practice, all of the Question and User features are available at the time of question creation, and it is only really the Answer features where ambiguity comes in. With the comments, for example, the norm is that comments lead to the refinement (via post-editing) of the answer, and the vast majority of comments in our dataset were posted soon after the original answer. Thus, while it is certainly possible for comments to appear after the answer has been finalised, any biasing effect here is minor. The only feature which has potentially changed significantly from the time of answer posting is the number of answer views, although as we will observe empirically, the utility of this feature is slight.

4.3 Combining Text and Metadata

To combine textual and metadata features, we concatenate $[\mathbf{v}_{\text{text}}; \mathbf{v}_{\text{meta}}]$ as the input question/answer pair embedding for the classification network H.

We define the prediction $\hat{y} := H([\mathbf{v}_{\text{text}}, \mathbf{v}_{\text{meta}}])$, where $\hat{y} \in \mathbb{R}^C$ in the case of $C = 2$ classes (i.e. "best" or not).

Now given training instance n, for the prediction $\hat{y}_c^{(n)}$ and true binary labels $y^{(n)} \in \{0, 1\}^C$, the training objective is the categorical cross-entropy loss $L = \frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} y_c^{(n)} \log \hat{y}_c^{(n)}$.

5 Experiments

To train our models, we used the Adam Optimiser (Kingma and Ba, 2014). For `decatt`, we used dropout over F, G, H after every dense layer. For the `doc2vec` MLP, we included batch normalisation before, and dropout after, each dense layer. For testing, we picked the best model according to the validation results after the end of each epoch.

The parameters for `decatt` were initialised with a Gaussian distribution of mean 0 and variance 0.01, and for the `doc2vec` MLP we used Glorot normal initialization. For Stack Overflow, the parameters for Word embeddings were pretrained using `GloVe` (Pennington et al., 2014) with the FULL data (by combining all questions and answers in the sequence they appeared) for 50 epochs. Word embeddings were set to 150 dimensions. The co-occurrence weighting function's maximum value $x_{\max}$ was kept at the default of 10. For SEMEVAL, we used pretrained Common Crawl cased embeddings with 840G tokens and 300 dimensions (Pennington et al., 2014).

To train the `decatt` model for Stack Overflow we split the data into 3 partitions based on the size of the question/answer text, with separate partitions where the total length of the question/answer text was: (1) ≤ 500 words; (2) > 500 and ≤ 2000 words; and (3) > 2000 words. We used a different batch size for each partition (32, 8, and 4 respectively).[6] Examples were shuffled within each partition after every epoch. For SEMEVAL we did not

[4] In total there were 86 badges in the dataset that users could obtain.

[5] For instance if a question has 4 tags, then the `QTags` feature group for that question has value 0.25 for the 4 tags present, and 0 for the other dimensions.

[6] This was to avoid running into memory issues when training with a large batch size on very long question and answer pairs.

Type	Name	Size	Description
Question	`BasicQ`	3	Number of times question has been viewed ($\times 1$), creation date of question ($\times 1$), and date of most recent activity on question ($\times 1$).
	`QTags`	n	Probability distribution over top-n tags for question ($\times n$).
Answer	`BasicA`	3	Number of times answer has been viewed ($\times 1$), creation date of answer ($\times 1$), and date of most recent activity on answer ($\times 1$).
	`Comments`	10	Number of comments on question and answer ($\times 2$), whether asker/answerer commented on question/answer ($\times 4$), number of sequential comments between asker and answerer across both question and answer ($\times 1$), average sentiment of comments on answer using Manning et al. (2014), both including and ignoring neutral sentences ($\times 2$), and whether there was at least one comment on answer ($\times 1$).
User	`BasicU`	8	Creation date of user account ($\times 1$), number of up/down votes received by user ($\times 2$), reputation value ($\times 1$), number questions asked/answered ($\times 2$), number of questions answered that were chosen as best ($\times 1$), number of comments made ($\times 1$).
	`Badges`	86	Whether user has each badge or not ($\times 86$).
	`UTags`	$2n$	Probability distribution over top-n tags across all questions answered by user ($\times n$), and the same distribution restricted to questions answered by the user where their answer was chosen as best ($\times n$).

Table 2: Summary of the metadata features used to improve question answering performance. These features are separated into feature groups, which in turn are separated into group types based on whether the values are specific to a given question, to a question's answer, or to a user.

use partitions, and instead used a batch size of 32, since training was fast enough.

For `doc2vec` pre-training, we used the FULL corpus, with `train`, `val` and `test` documents excluded.[7] We used the `dbow` version of `doc2vec`, and included an additional `word2vec` step to learn the word embeddings simultaneously.[8]

Note that for SEMEVAL, we experiment with using only the text features to better understand the competitiveness of these text-processing networks `decatt` and `doc2vec`.

We tuned hyperparameters for all methods based on validation performance using the SigOpt Bayesian optimisation service. Optimal hyperparameter configurations are detailed in Table 3.

For additional comparison, we implemented the following baselines (some taken from Jansen et al. (2014), plus some additional baselines of our own), including: (1) `random`, which ranks the answers randomly; (2) `first-answer`, which ranks the answers in chronological order; (3) `highest-rep`, which ranks the answers by decreasing reputation; (4) `longest-doc`, which ranks the answers by decreasing length; and (5) `tf-idf`, which ranks the answers by the cosine of the tf-idf[9] vector representations between the

[7]The text was additionally preprocessed by lowercasing. `doc2vec` training and inference was done using the `gensim` (Řehůřek and Sojka, 2010) implementation.

[8]Based on Lau and Baldwin (2016), our hyperparameter configuration of `doc2vec` for training was as follows: vector size $= 200$; negative samples $= 5$; window size $= 3$; minimum word frequency $= 5$; frequent word sampling threshold $= 1 \cdot 10^{-5}$; starting learning rate (α_{start}) $= 0.05$; minimum learning rate ($\alpha_{\min}$) $= 0.0001$; and number of epochs $= 20$. For inferring vectors in our `train`, `val` and `test` sets we used: $\alpha_{\text{start}} = 0.01$; $\alpha_{\min} = 0.0001$; and number of epochs $= 500$.

[9]Generated based on the training partition.

Model	Dataset	Hyperparameter					
		F	G	H	Tags	Dropout	LR
decatt + metadata	SMALL	$188, 127$	$110, 110$	$282, 32$	0	0.68	$2.5 \cdot 10^{-6}$
	LARGE	$500, 179$	$221, 221$	$533, 523$	24	0.49	$6.4 \cdot 10^{-5}$
doc2vec + metadata	SMALL	N/A	N/A	$1000, 92, 194$	0	0.90	$2 \cdot 10^{-3}$
	LARGE	N/A	N/A	$1000, 212, 968$	25	0.77	$6 \cdot 10^{-4}$
metadata	SMALL	N/A	N/A	$50, 50$	0	0.65	$3 \cdot 10^{-3}$
	LARGE	N/A	N/A	$555, 600$	24	0.65	$8 \cdot 10^{-3}$
decatt	SMALL	$188, 127$	$110, 110$	$145, 500$	N/A	0.68	$1.7 \cdot 10^{-4}$
	LARGE	$500, 179$	$221, 221$	$53, 44$	N/A	0.37	$3.1 \cdot 10^{-5}$
	SEMEVAL	$200, 200$	$200, 200$	$200, 200$	N/A	0.5	$5 \cdot 10^{-4}$
doc2vec	SMALL	N/A	N/A	$791, 737, 414$	N/A	0.56	$6 \cdot 10^{-5}$
	LARGE	N/A	N/A	$1000, 558, 725$	N/A	0.56	$3 \cdot 10^{-5}$

Table 3: Hyperparameter settings used for each model and corpora. "LR" = learning rate; "N/A" indicates that the hyperparameter is not relevant for the given model. All models were trained for 40 epochs.

Model	SMALL	LARGE	SEMEVAL
decatt + metadata	**.432**	**.527**	N/A
doc2vec + metadata	.429	.513	N/A
metadata	.403	.463	N/A
decatt	.346	.363	.865
doc2vec	.343	.353	.740
random	.185	.185	.618
first-answer	.234	.243	.726
tf-idf	.245	.246	.647
highest-rep	.271	.268	N/A
longest-doc	.318	.337	.720
semeval-best	N/A	N/A	**.884**

Table 4: Results for `doc2vec`, `metadata` and `decatt` models on both Stack Overflow datasets (P@1) and SEMEVAL (MAP).

Model	SMALL	LARGE
All features	.403	.496
−BasicQ	.399 (−.004)	.495 (−.001)
−QTags	.403 (−.000)	.442 (−.054)
−BasicA	.400 (−.003)	.497 (+.001)
−Comments	.303 (−.100)	.410 (−.086)
−BasicU	.394 (−.009)	.485 (−.011)
−Badges	.408 (+.005)	.499 (+.003)
−UTags	.403 (−.000)	.433 (−.063)

Table 5: Feature ablation results for the `metadata` model, based on fixed hyperparameter settings (P@1).

question and answer.

5.1 Results

For a given question, we are interested both in how accurately our model ranks the answers, and whether it classifies the best answer correctly. However, for simplicity we simply look at the performance of the model in correctly predicting the best answer. Following Bogdanova and Foster (2016), we measure this using P@1. In all cases this is calculated on the `test` set of questions, using the gold-standard "best answer" labels from the Stack Overflow corpus, as decided by the question asker. For SEMEVAL we use MAP to compare with other published results. The results are presented in Table 4. To investigate the relative importance of the different metadata feature groups, we additionally provide feature ablation results in Table 5 for the Stack Overflow dataset.

We can make several observations from these results. Firstly, we can see that performance increases when we increase the dataset size (from SMALL to LARGE), showing that our models scale well with more data. For the text-only models, `decatt` outperforms `doc2vec` consistently over both datasets. In addition, `metadata` achieves much higher results than the text-only models, which shows the importance of utilising the rich metadata data available for cQA retrieval. The best model, `decatt + metadata`, is the hybrid model that combines both sources of information and substantially improves performance compared to `metadata`. From the SEMEVAL results, we can see that our best text model (`decatt`) is competitive with the state-of-the-art

Question		Best answer		Incorrect answer	
Q_1:	... What I'd love to hear is what you specifically use at your company ... 1. How do users report bugs/feature requests to you? What software do you use to keep track of them? 2. How do bugs/feature requests get turned into "work"? ...	$A_{1,1}$:	For my (small) company: We design the UI first. ... As we move towards an acceptable UI, we then write a paper spec for the workflow logic of the application ...	$A_{1,2}$:	To give a better answer, my company's policy is to use XP as much as possible and to follow the principles and practices as outlined in the Agile manifesto. ...
Q_2:	I am trying to pass a pointer `rgb` that is initialized with `memset` to 0 and then looped through to place a 32 bit integer only in the bounds ...	$A_{2,1}$:	I went through and basically sorted out all the warts, and explained why. A lot of it amounts to the fact that if your compiler spits out warnings, you have to listen to them. ...	$A_{2,2}$:	This actually has a number of bugs, but your first issue is assigning the pixel value to the array ... You don't need to reset `j` to 0 ... Also, you're misusing `sizeof()` ...

Table 6: Example questions and answers that were misclassified by one of `decatt` or `metadata`.

model (`semeval-best`), which also incorporates a number of handcrafted metadata features to achieve a score of 88.4%.

To better understand the attention learnt by `decatt`, we plotted the attention weights for a number of question–answer pairs in Figure 1. In general, technical words that appear relevant to the question and answer have a high weight. Overall, we find that `decatt` does not appear to capture word pairs which correspond to each other, as important question words are given strong attention consistently for most answer words. We do find a few exceptions with strong mutual attention, e.g. *roughly 10-20+ connections* and *multiple concurrent sockets* have strong mutual attention. This may explain the small difference in performance between the `doc2vec` and `decatt` models.

In terms of our feature ablation results, all feature types contribute to an increase in performance. The increases are greater in LARGE, suggesting that the model is better able to utilize the information given more data. The `BasicQ`, `BasicA` features, which include dates and view counts, do not appear to be of much use. Niether does `Badges`, which appears to hurt the model slightly. The other features give substantial gains, especially in LARGE. The `Comments` feature is strongest, but since it includes information based on the comments of the question asker, it may not be as relevant for the ultimate goal of cross-question answer retrieval.

Comparing `decatt` and `metadata` model, we found that overall, both models perform well, and even when a model does not predict the accepted answer it often gives a highly-voted answer. We found that the `metadata` model tends to favour answers which have multiple comments involving the asker, and especially answers from high-reputation users. For example, in answer $A_{1,2}$ to question Q_1 in Table 6, there were a total of 8 comments to the answer (and no comments to any of the other answers), biasing `metadata` to prefer it. In practice, however, those comments were uniformly negative on the part of a number of prominent community members, which the model has failed to capture. This makes sense given the results in Table 5. However, it does not appear to understand comments where the asker is discussing why the answer fails to address his question, for example *I can't choose one Polygon class because each library operates only in its own implementation*. While we include sentiment features in our metadata features, this alone might not be sufficient, since the disussion may revolve around facts and require more detailed modelling of the discourse structure of comments. Note that here, `decatt` correctly selected $A_{1,1}$, on the basis of its content.

As an example of a misclassification by `decatt`, answer $A_{2,2}$ is preferred over (best-answer) $A_{2,1}$ in response to question Q_2 in Table 6, but is actually a more comprehensive answer which deals with more issues in the original code and receives an equal number of community votes from the community to $A_{2,2}$. However, $A_{2,1}$ was posted first and receives a comment of gratitude from the question asker, meaning that `metadata` is able to correctly classify it as best answer.

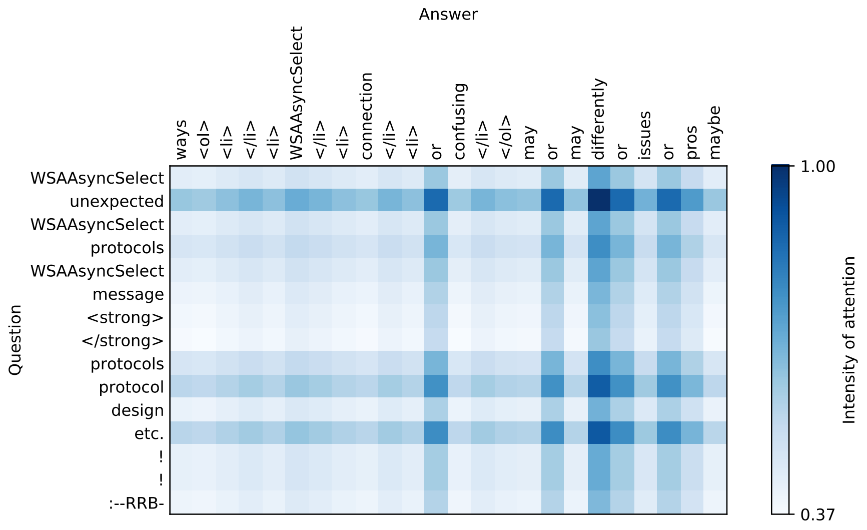

Figure 1: Attention weights for a question–answer pair, e, normalized to $[0, 1]$. Due to the long length of question and answers we only plot weights above some threshold.

5.2 Future Work

There are multiple avenues for future research based on our work. Our model's use of attention in the Stack Overflow dataset appears to be very limited, so a model which can make full use of attention could be a good direction of investigation. Another approach would be to extend our model to incorporate the entire list of answers and comments, possibly using graph-based approaches, instead of relying on individual question/answer pairs and manually engineered comment features. Ultimately, we would like to extend our methodology for cross-question answer retrieval, rather than just answer retrieval from a single question, given the goal of utilising the data in cQA forums to facilitate general-purpose non-factoid question answering

6 Conclusions

In this paper we built a state-of-the-art model for cQA answer retrieval model based on a deep-learning framework. Unlike recent work on this problem we successfully utilised metadata to substantially boost performance. In addition, we adapt an attentional component in our model, which improves results over the simple paragraph vector-based approach used in our benchmark, which was previously the state-of-the-art model. It is our hope that this work facilitates future research on utilising cQA data for non-factoid question answering.

References

Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding high-quality content in social media. In *Proceedings of the 1st ACM International Conference on Web Search and Data Mining (WSDM)*, pages 183–194.

Xin-Qi Bao and Yun-Fang Wu. 2016. A tensor neural network with layerwise pretraining: Towards effective answer retrieval. *Journal of Computer Science and Technology*, 31(6):1151–1160.

Yonatan Belinkov, Mitra Mohtarami, Scott Cyphers, and James Glass. 2015. VectorSLU: A continuous word vector approach to answer selection in community question answering systems. In *Proceedings of the 9th Conference on Semantic Evaluation (SemEval)*.

Dasha Bogdanova and Jennifer Foster. 2016. This is how we do it: Answer reranking for open-domain how questions with paragraph vectors and minimal feature engineering. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 16)*, San Diego, USA.

R. Samuel Bowman, Gabor Angeli, Christopher Potts, and D. Christopher Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 632–642, Lisbon, Portugal.

Grégoire Burel, Yulan He, and Harith Alani. 2012. Automatic identification of best answers in online enquiry communities. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, pages 514–529.

Pedro Chahuara, Thomas Lampert, and Pierre Gancarski. 2016. Retrieving and ranking similar questions from question-answer archives using topic modelling and topic distribution regression. In *Proceedings of the 20th International Conference on Theory and Practice of Digital Libraries (TPDL): Research and Advanced Rechnology for Digital Libraries*, pages 41–53.

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Peter Jansen, Mihai Surdeanu, and Peter Clark. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 977–986.

Jiwoon Jeon, W Bruce Croft, Joon Ho Lee, and Soyeon Park. 2006. A framework to predict the quality of answers with non-textual features. In *Proceedings of the 29th International Conference on Research and Development in Information Retrieval (SIGIR 2006)*, pages 228–235.

Shafiq Joty, Giovanni Da, Llúis Màrquez, and Preslav Nakov. 2016. Joint learning with global inference for comment classification in community question answering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL HLT 2016)*, pages 703–713.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Yuta Koreeda, Takuya Hashito, Yoshiki Niwa, Misa Sato, Toshihiko Yanase, Kenzo Kurotsuchi, and Kohsuke Yanai. 2017. Bunji at SemEval-2017 task 3: Combination of neural similarity features and comment plausibility features. In *Proceedings of the International Workshop on Semantic Evaluation*, pages 353–359, Vancouver, Canada.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 78–86, Berlin, Germany.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *arXiv Preprint arXiv:1405.4053*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014): System Demonstrations*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Preslav Nakov, Doris Hoogeveen, Llúis Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 Task 3: Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 27–48, Vancouver, Canada.

Henry Nassif, Mitra Mohtarami, and James Glass. 2016. Learning semantic relatedness in community question answering using neural models. In *Proceedings of the 1st Workshop on Representation Learning for NLP (RepL4NLP)*, pages 137–147.

Adi Omari, David Carmel, Oleg Rokhlenko, and Idan Szpektor. 2016. Novelty based ranking of human answers for community questions. In *Proceedings of the 39th International Conference on Research and Development in Information Retrieval (SIGIR 2016)*, pages 215–224.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *arXiv:1606.01933 [Cs]*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta.

Chirag Shah. 2015. Building a parsimonious model for identifying best answers using interaction history in community Q&A. *JASIST*, 52(1):1–10.

Anna Shtok, Gideon Dror, Yoelle Maarek, and Idan Szpektor. 2012. Learning from the past: Answering new questions with past answers. In *Proceedings of the 21st International Conference on the World Wide Web (WWW 2012)*, pages 759–768.

Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. 2017. Neural paraphrase identification of questions with noisy pretraining. In *arXiv:1704.04565 [Cs]*.

Xudong Tu, Xin-Jing Wang, Dan Feng, and Lei Zhang. 2009. Ranking community answers via analogical reasoning. In *Proceedings of the 18th International World Wide Web Conference*, pages 1227–1228.

Jelica Vasiljevic, Tom Lampert, and Milos Ivanovic. 2016. The application of the topic modeling to question answer retrieval. In *Proceedings of the 6th International Conference of Information Society and Technology (ICIST)*, pages 241–246.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103.

Baoxun Wang, Xiaolong Wang, Chengjie Sun, Bingquan Liu, and Lin Sun. 2010. Modeling semantic relevance for question-answer pairs in web social communities. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 1230–1238.

Xin-Jing Wang, Xudong Tu, Dan Feng, and Lei Zhang. 2009. Ranking community answers by modeling question-answer relationships via analogical reasoning. In *Proceedings of the 32nd International Conference on Research and Development in Information Retrieval (SIGIR 2009)*, pages 179–186.

Xiaobing Xue, Jiwoon Jeon, and W Bruce Croft. 2008. Retrieval models for question and answer archives. In *Proceedings of the 31st International Conference on Research and Development in Information Retrieval (SIGIR 2008)*, pages 475–482.

Kai Zhang, Wei Wu, Haocheng Wu, Zhoujun Li, and Ming Zhou. 2014. Question retrieval with high quality answers in community question answering. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management (CIKM)*, pages 371–380.

Guangyou Zhou, Yin Zhou, Tingting He, and Wensheng Wu. 2016. Learning semantic representation with neural networks for community question answering retrieval. *Knowledge-Based Systems*, 93:75–83.

Zainab Zolaktaf, Fatemeh Riahi, Mahdi Shafiei, and Evangelos Milios. 2011. Modeling community question-answering archives. In *Proceedings of the 2nd Workshop on Computational Social Science and the Wisdom of Crowds*, pages 1–5.