AVA: an Automatic eValuation Approach for Question Answering Systems

Thuy Vu
Amazon Alexa AI
Manhattan Beach, CA, USA
thuyvu@amazon.com

Alessandro Moschitti
Amazon Alexa AI
Manhattan Beach, CA, USA
amosch@amazon.com

Abstract

We introduce AVA, an automatic evaluation approach for Question Answering, which given a set of questions associated with Gold Standard answers (references), can estimate system Accuracy. AVA uses Transformer-based language models to encode question, answer, and reference texts. This allows for effectively assessing answer correctness using similarity between the reference and an automatic answer, biased towards the question semantics. To design, train, and test AVA, we built multiple large training, development, and test sets on public and industrial benchmarks. Our innovative solutions achieve up to 74.7% F1 score in predicting human judgment for single answers. Additionally, AVA can be used to evaluate the overall system Accuracy with an error lower than 7% at 95% of confidence when measured on several QA systems.

1 Introduction

Accuracy evaluation is essential both to guide system development as well as to estimate its quality, which is important for researchers, developers, and users. This is often conducted using benchmark datasets containing a data sample, *possibly* representative of the target data distribution, provided with Gold Standard (GS) labels (typically produced with a human annotation process). The evaluation is done by comparing the system output with the expected labels using some metrics.

This approach falls short when the system output spans a large, possibly infinite set of correct items. For example, in retrieval-based Question Answering (QA) systems, a correct answer can be any string in the referent text database. For example, for the question, *When did Marlins start?*, an answer could be: *The Miami Marlins began play in the 1993 season as the Florida Marlins*; *They started in 1993*; *They firstly played in 1993*; *In 1993*; or any possible natural language text conveying the information that they started in 1993. As annotating all possible system pieces of output is infeasible, the standard approach is to re-evaluate the new output of the system manually. This dramatically limits the experimentation velocity while significantly increases the development costs.

A viable solution for specific NLP tasks such as Machine Translation (MT), automatically estimates an evaluation score between the system and the reference answers, which correlates with human judgment, e.g., the BLEU score is one popular measure (Papineni et al., 2002). Such methods cannot be applied to a standard QA setting, since QA systems, e.g., those developed for TREC-QA track (Voorhees and Tice, 1999), have the purpose to provide correct answers and are evaluated with Accuracy, i.e., the percentage of correct answers. Segment overlapping metrics such as BLEU, METEOR, or ROUGE do not provide a binary outcome, i.e., correct or incorrect (as this is not the aim of MT evaluation).

Hypothetically speaking, we could apply a threshold to their score to obtain a binary outcome. However, it would not be sufficient as the correctness of an answer loosely depends on the match between the reference and candidate answers. Two answers can be correct or incorrect independently of their overlap with the reference. For example, for the question, *What percentage of water in the body?*, associated with a reference, *The percentage of water in the body is 60%*, a correct answer is *Most of the human body is water, with an average of roughly 60%*. In contrast, an incorrect answer, still very similar to the reference, could be: *The percentage of water in the body is variable*. The MT metrics above would find the similarity of the reference with the incorrect answer higher than the one of the references with the correct answer. Even a powerful model such as BERTScore (Zhang et al., 2020) would not provide a higher score to the correct answer since it is an unsupervised approach, not trained for this task.

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5223–5233
June 6–11, 2021.

It should also be noted that simply training models for matching the answer candidate with the reference will again not work. The question semantics would radically influence the correctness of the answer. That is, $match(t, r|q_1)$ can be true while $match(t, r|q_2)$ can be false, where t and r are a pair of answer candidate and reference, and q_1 and q_2 are two different questions.

In this paper, we study the design of models for measuring the Accuracy of QA systems, i.e., percentage of correct answers over a test set (to our knowledge this is the first successful and thorough study). In particular, we (i) build several baselines based on pre-trained Transformer models (Devlin et al., 2019; Liu et al., 2019) to encode the triple, question q, candidate t, and reference r, in different ways; and (ii) propose a new attention mechanism, peer attention, to model the interaction between t and r, given the semantic bias of q.

To develop and test our models, we created (i) a dataset, Web-based Question Answering[1] (WQA) for training and testing AVA, the point-wise estimation of QA system output, i.e., the evaluation if an answer is correct or not, given a GS answer; and (ii) a System Dataset (SD) constituted by a set of outputs from several QA systems, for which AVA estimates their Accuracy.

The results show a high F1 for point-wise models, up to 74.7%. AVA can almost always rank systems in terms of Accuracy as manual annotation does. Finally, the Root Mean Square Error (RMSE) with respect to human evaluation depends on the datasets, ranging from 2% to 9.5%, with a Std. Dev. lower than 5%.

2 Related Work

Automatic evaluation has been an interesting research area for decades (Papineni et al., 2002; Magnini et al., 2002). There are two typical strategies to design an automatic evaluator: supervised and unsupervised. In MT research, for example, BLEU (Papineni et al., 2002) has been a very popular unsupervised evaluation method for the task. Other supervised methods have been recently proposed, most notably (Ma et al., 2019). Neural-based automatic evaluators for dialog systems were studied in (Ghazarian et al., 2019; Lowe et al., 2017; Tao et al., 2017; Kannan and Vinyals, 2017).

Automatic evaluation for QA was addressed by Magnini et al. (2002) and also for multiple sub-domain QA systems (Leidner and Callison-Burch, 2003; Lin and Demner-Fushman, 2006; Shah and Pomerantz, 2010; Gunawardena et al., 2015). However, little progress has been made in the past two decades towards obtaining a standard method. Automating QA evaluation is still an open problem, and there is no recent work supporting it. As mentioned in the introduction MT unsupervised metrics, e.g., BLEU score or BERTScore, are not either a solution or a reasonable baseline for automatic QA evaluation. They could be used as features for our models, but we designed several supervised approaches based on pre-trained Transformer models, which subsume these MT features.

A remotely related research effort for automatizing answer evaluation concerns student essays. Short answer grading (SAG), or short answer scoring, involves the automatic grading of students' answers, typically written in free text, for a given prompt or question (Mohler et al., 2011). This task has been studied in (Mitchell et al., 2002; Pulman and Sukkarieh, 2005) for educational applications. Neural-based systems have also been recently proposed to improve the models (Riordan et al., 2017; Wang et al., 2019). Despite the conceptual similarity, i.e., evaluating an answer, the problem setting for the task is fundamentally different.

Specifically, SAG is prompt-centric; thus, the learning objective is to score accurately other different answer variants for a particular question by building models trained on previously known variants (Wang et al., 2019). Besides, the answers, while written in free text, are not typically complete sentences. Therefore, the SAG design aims to capture sufficient content covered in the reference responses for a question. On the contrary, AVA is designed to operate in an open-domain QA setting, where both the question and answer are arbitrary input and complete sentences.

3 Problem definition and preliminaries

We consider retrieval-based QA systems, which are mainly constituted by (i) a search engine, retrieving top-k documents related to the questions, and (ii) an Answer Sentence Selection (AS2) model, which reranks passages/sentences extracted from the documents. We can automatically evaluate the (i) Accuracy of the QA system, which is the percentage of correct top sentences, and (ii) complex measures, such as MAP and MRR, which quantify the quality of the rank produced by the AS2 model.

[1] Available at github.com/alexa/wqa_ava

q:	What is the population of California?
r:	With slightly more than 39 million people (according to 2016 estimates), California is the nation's most populous state—its population is almost one and a half times that of second-place Texas (28 million).
s:	39 million
t:	The resident population of California has been steadily increasing over the past few decades and has increased to 39.56 million people in 2018.

Table 1: An example of input data

3.1 Answer Sentence Selection (AS2)

The task of reranking answer sentence candidates provided by a retrieval engine can be modeled with a classifier scoring the candidates. Let q be a question, $T_q = \{t_1, \ldots, t_n\}$ be a set of answer sentence candidates for q, we define $\mathcal{R}$ as a ranking function, which orders the candidates in T_q according to a score, $p\,(q, t_i)$, indicating the probability of t_i to be a correct answer for q. Popular methods modeling $\mathcal{R}$ include Compare-Aggregate (Yoon et al., 2019), inter-weighted alignment networks (Shen et al., 2017), and Transformers (Garg et al., 2020).

3.2 Automatic evaluation of QA

The AVA performance can be measured in two ways: (i) evaluation of the single answers provided by the target system (point-wise evaluation); and (ii) the aggregated evaluation of a set of questions (system-wise evaluation). We define the former as a function: $\mathcal{A}\,(q, r, t_i) \rightarrow \{0, 1\}$, where r is a reference answer (from GS) and the output is simply a correct/incorrect label. Table 1 shows an example question associated with a reference, a system answer, and a short answer s[2].

$\mathcal{A}$ can be applied to compute the final Accuracy of a system using an aggregator function: we simply assume the point-wise AVA predictions as they were the GS. For example, in case of Accuracy, we simply average the AVA predictions, i.e., $\frac{1}{|Q|}\sum_{q \in Q} \mathcal{A}(q, r, t[, s])$, where s is a short GS answer (e.g., used in machine reading). It is an optional input, which we only use for building a linear model baseline, described in Section 5.

4 Dataset creation

To learn and test our models, we needed to build AVA datasets. The interesting aspect is that we can automatically derive them from standard AS2 corpora if they contain questions with multiple correct answers. For this purpose, we created our dataset WQA for AS2 and transformed it into AVA-WQA. We describe our approach to transforming AS2 to AVA datasets in this section. Finally, we build another benchmarking dataset for AVA constituted by a set of QA systems and their output on target test sets. This is used to measure the end-to-end system performance (system-wise evaluation).

4.1 AS2 datasets

These datasets consist of a set of questions Q, and for each $q \in Q$, there are $T_q = \{t_1, \ldots, t_n\}$ candidates, comprised of both correct answers C_q and incorrect answers $\overline{C_q}$, $T_q = C_q \cup \overline{C_q}$.

WQA: The Web-based Question Answering is a dataset built by Alexa AI as part of the effort to improve understanding and benchmarking in QA systems. The creation process includes the following steps: (i) given a set of questions we collected from the web, a search engine is used to retrieve up to 1,000 web pages from an index containing hundreds of millions of pages. (ii) From the retrieved documents, all candidate sentences are extracted and ranked using AS2 models from (Garg et al., 2020). Finally, (iii) top candidates for each question are manually assessed as correct or incorrect by human judges. This allowed us to obtain a richer variety of answers from multiple sources with a higher average number of answers, as shown in Table 2.

4.2 Point-wise datasets for AVA

We use AS2 datasets as follows: firstly, we only keep questions with at least two correct answers, which is critical to build positive and negative examples. Secondly, given $\langle q, t_i, t_j \rangle$, where t_i, t_j are two candidates, we build:

AVA-Pos $= \langle q, (t_i, t_j) \in C_q \times C_q \text{ and } t_i \neq t_j \rangle$
AVA-Neg $= \langle q; (t_i, t_j) \in C_q \times \overline{C_q} \rangle$

We create AVA-WQA from WQA. The statistics are shown in Table 2.

4.3 AVA System Dataset (SD)

To measure AVA with respect to the overall system Accuracy, we need to have a sample of systems and their output on different test sets. We created a dataset with candidate answers collected from eight systems answering a set of 1,340 questions. The questions were again sampled from the Web. We only considered information questions.

[2]The latter can be very effective but it adds an additional annotation cost, thus we limit its use just for the baseline model. That is, we aim to have a lower cost AVA model.

	WQA			WQA Qs with multiple As			AVA-WQA		
data split	#Qs	#As	#wrong-As	#Qs	#As	#wrong-As	positives	negatives	total
Train	262	5,399	20,801	245	5,382	20,748	183,894	349,765	533,659
Dev.	283	8,682	19,618	276	8,674	19,502	430,230	426,246	856,476
Test	294	9,412	19,988	281	9,399	19,790	479,028	449,625	928,653

Table 2: WQA and AVA-WQA Statistics

The systems differ from each other in multiple ways including: (i) *modeling*: Compare-Aggregate (CNN-based) and different Transformers-based architectures with different hyper-parameter settings; (ii) *training*: the systems are trained on different resources; (iii) *candidates*: the pool of candidates is collected and filtered differently and in different numbers; and (iv) *retrieval*: different search engines, diverse indexed data sources, and different retrieval settings. This system variability provides high generality of our AVA results.

5 Models for AVA

The central intuition for the design of an automatic QA evaluator is (i) capturing the same information a standard QA system uses, while (ii) exploiting the semantic similarity between t and r, biased by q. We build three types of models: (i) a linear classifier, which is more interpretable and can help the model design, (ii) Transformer-based methods, based on powerful language models, and (iii) our Peer Attention approach to better model the interaction among q, t, and r.

5.1 A linear classifier

Given an input example, (q, r, s, t), our classifier uses the following similarity features: x_1=*is-included*(s, t), x_2=*sim-text*(r, t), x_3=*sim-text*(r, q); and x_4=*sim-text*(q, t), where *is-included* applied to s and t is a binary feature testing if t includes s, *sim-text* is a sort of Jaccard similarity defined as: $\textit{sim-text}\,(s_i, s_j) = 2\frac{|\textit{tok}(s_i)\cap\textit{tok}(s_j)|}{|\textit{tok}(s_i)|+|\textit{tok}(s_j)|}$, and $\textit{tok}\,(s)$ is a function that splits s into tokens.

Let $\mathbf{x} = f\,(q, r, s, t) = (x_1, x_2, x_3, x_4)$ be a similarity feature vector describing our evaluation tuple, and let l be a binary label indicating whether t answers q or not. We train $\mathbf{w}$ on a dataset $D = \{(\mathbf{x}_i, l_i)\}$, $i = 1, .., |D|$, using SVM. We compute the point-wise evaluation of t as the test $\mathbf{x}_i \cdot \mathbf{w} > \alpha$, where α is a threshold trading off Precision for Recall in standard classification approaches.

5.2 Transformer-based models

Transformer-based architectures have delivered powerful language models, which can capture complex similarity patterns. Thus, they are suitable methods to improve our basic approach described in the previous section. Following the linear classifier modeling, we propose three different ways to exploit the relations among the members of the tuple (q, r, s, t).

Let $\mathcal{B}$ be a pre-trained language model, e.g., the recently proposed BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), XLNet (Yang et al., 2019), AlBERT (Lan et al., 2020). We use $\mathcal{B}$ to compute the embedding representation of a tuple: $\mathcal{B}\,(a, a') \rightarrow \mathbf{x} \in \mathbb{R}^d$, where (a, a') is a short text pair, $\mathbf{x}$ is the output representation of the pair, and d is the dimension of the output representation. We use a standard feedforward network, i.e., $\mathcal{A}\,(\mathbf{x}) = \mathbf{W}^\intercal\mathbf{x} + b$, to implement the classification layer, deciding if an answer is correct, where $\mathbf{W}$ and b are parameters we learn by fine-tuning the model on AVA datasets. We describe the following different designs for $\mathcal{A}$.

$\mathcal{A}_0$: Text-pair embedding

We build a language model representation for pairs of members of the tuple, $x = (q, r, t)$ by simply inputting them to Transformer models $\mathcal{B}$ in the standard sentence pair fashion. We consider four different configurations of $\mathcal{A}_0$, one for each of the following pairs: (q, r), (q, t), (r, t), and one for the triplet, (q, r, t), modeled as the concatenation of the previous three representations. The representation for each pair is produced by a different and independent Transformer instance, i.e., $\mathcal{B}_p$. More formally, we have the following three models $\mathcal{A}_0\,(\mathcal{B}_p)$, $\forall p \in \mathcal{P}_0$, where $\mathcal{P}_0 = \{(q, r), (q, t), (r, t)\}$. Additionally, we design a model over (q, r, t) with $\mathcal{A}_0\,(\cup_{p\in\mathcal{P}_0}\,\mathcal{B}_p)$, where $\cup$ means concatenation of the representations. We do not use the short answer, s, as its contribution is minimal when using powerful Transformer-based models.

$\mathcal{A}_1$: Improved text-triple embedding

The methods above are limited to pair representations. We improve them by designing $\mathcal{B}$ models that can capture pattern dependencies across q, r and t. To achieve this, we concatenate pairs of the three pieces of text. We indicate this string concatenation with the $\circ$ operator. Specifically, we consider $\mathcal{P}_1 = \{(q, r \circ t), (r, q \circ t), (t, q \circ r)\}$ and propose the following $\mathcal{A}_1$. As before, we have the individual models, $\mathcal{A}_1\,(\mathcal{B}_p), \forall p \in \mathcal{P}_1$ as well as the combined model, $\mathcal{A}_1\,(\cup_{p \in \mathcal{P}_1}\ \mathcal{B}_p)$, where again, B_p uses different instances that are fine-tuned together.

$\mathcal{A}_2$: Peer Attention for Transformer models

Our previous designs instantiate different $\mathcal{B}$ for each pair; thus, they learn the feature representations of the target pair and the relations between its members during the fine-tuning process. This individual optimization limits the modeling of patterns across the representations of different pairs as there is no attention mechanism between the $\mathcal{B}$ instances: the combination of features *only* happens in the last classification layer.

We propose *Peer Attention* to improve feature connections between different $\mathcal{B}$ instances. The idea, similar to the encoder-decoder setting in Transformer-based models (Vaswani et al., 2017), is to introduce an additional decoding step for each pair. That is, we use another Transformer instance to decode the output from the previous instance.

Figure 1 depicts our proposed setting for learning the representation of two different pairs: a (e.g., equal to (q, t)) and b (e.g., equal to (q, r)). The approaches from the previous section would learn two Transformer instances, $\mathcal{B}_a$ and $\mathcal{B}_b$, with one pass. Our *Peer Attention*, instead, operates two steps, using four instances, $\mathcal{B}_{a_0}$, $\mathcal{B}_{a_1}$, $\mathcal{B}_{b_0}$, and $\mathcal{B}_{b_1}$ as follows: First, in the encoding step, we learn the representations, $\mathcal{B}_{a0}$ and $\mathcal{B}_{b0}$, as before. Second, in the decoding step, we use the $H_{[CLS]_{a_0}}$ from $\mathcal{B}_{a_0}$ and $H_{[CLS]_{b_0}}$ from $\mathcal{B}_{b_0}$, and concatenate them to a and b, respectively, providing input to $\mathcal{B}_{a_1}$ and $\mathcal{B}_{b_1}$ for the second pass of fine-tuning.

Thus, the representation in one pair can attend over the representation in the other pair during the decoding stage. This allows the feature representations from each instance $\mathcal{B}$ to be shared during training and prediction stages. The final representation input to the classification layers is constituted by $H_{[CLS]_{a_0}}$, $H_{[CLS]_{a_1}}$, $H_{[CLS]_{b_0}}$, and $H_{[CLS]_{b_1}}$.

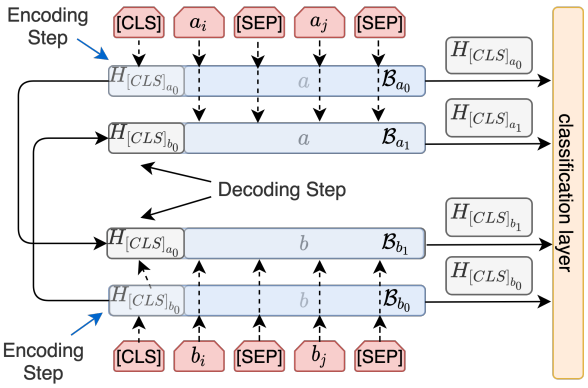

Figure 1: Peer attention on a and b pairs.

6 Experiments

We study the performance of AVA in predicting: (i) the correctness of the individual answers output by a system (point-wise estimation); and (ii) the overall system performance derived on a test set. We consider QA Accuracy and passage reranking measures in comparison to human labeling. The first aspect evaluates the quality of our approaches, whereas the second provides evidence on the practical use of AVA to develop QA systems.

6.1 Datasets and models

We train and test models using our new AVA-WQA dataset. We also evaluate the point-wise performance on the WikiQA and TREC-QA datasets.

Table 3 summarizes the configurations we consider for training and testing. As the linear classifier baseline, we used SVM by scikit-learn, setting the probability parameter to enable Platt scaling calibration on the classifier score.

We developed our Transformer-based AVA on top of the HuggingFace's Transformer library (Wolf et al., 2020), which also offers a native encoder-decoder setting through the `encoder_hidden_states` feature. We use RoBERTa-Base as the initial pre-trained model for each $\mathcal{B}$ instance (Liu et al., 2019), with the default hyper-parameter setting of GLUE trainings: (i) AdamW variant (Loshchilov and Hutter, 2017) as optimizer, (ii) a learning rate of $1e$-06 set for all fine-tuning exercises, and (iii) a maximum sequence length set to 128. Our number of iterations is two. We also use a development set to enable early stopping based on F1 measure after the first iteration. We fix the same batch size setting in the experiments to avoid possible performance discrepancies caused by different batch sizes.

Model Setting	Configurations
Linear Classifier	using 4 features x_i
$\mathcal{A}_0$	one for each and one for all from $\mathcal{P}_0$
$\mathcal{A}_1$	all possible combinations from $\mathcal{P}_1$
$\mathcal{A}_2$	the best setting from $\mathcal{A}_1$

Table 3: The AVA configurations used in training

6.2 Metrics

We study the performance of AVA in evaluating passage reranker systems, which differ not only in methods but also in domains and application settings. We employ the following evaluation strategies to benchmark AVA.

Point-wise evaluation We use Precision, Recall, and F1, to measure the performance of AVA in predicting if an answer candidate is correct or not.

System-wise evaluation We use AVA in a simple aggregator to estimate the overall system performance over a test set. The metrics we consider in our estimation are: Precision-at-1 (P@1), Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR), as TREC-QA and WikiQA contain answer ranks. In contrast, we only use P@1 on SD dataset, as this only includes the selected answers for each system.

To measure the quality of AVA with respect to GS annotation we use (i) Root Mean Square Error: $\text{RMSE}\,(a,h) = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(a_i - h_i)^2}$, where a and h are the measures given by AVA and the human annotation, respectively; and (ii) Kendall's Tau-b[3] to measure the correlation between the system ranks produced by AVA and GS one, i.e., $\tau = \frac{c-d}{c+d}$, where c and d are the numbers of concordant and discordant pairs between the two rankings.

6.3 Results on Point-wise Evaluation

We evaluate the performance of AVA in predicting if an answer t is correct for a question q, given a reference r. Table 4 shows the result. The first column reports the names of the systems described in Section 5. The second column shows the F1 measured on AVA-WQA. We note that:

- The SVM classifier performs much lower than any Transformer-based model (fed with a complete input): clearly, Transformer models can exploit powerful language models, suggesting that generalization is important.

[3]We use `scipy.stats.kendalltau`

Modeling configuration	F1
Linear Classifier	0.3999
$\mathcal{A}_0\,(\{(q,r)\})$	0.0695
$\mathcal{A}_0\,(\{(r,t)\})$	0.6247
$\mathcal{A}_0\,(\{(q,t)\})$	0.6713
$\mathcal{A}_0\,(\mathcal{P}_0)$	**0.6807**
$\mathcal{A}_1\,(\{(q,r\circ t)\})$	0.7014
$\mathcal{A}_1\,(\{(r,q\circ t)\})$	0.7383
$\mathcal{A}_1\,(\{(t,q\circ r)\})$	0.7236
$\mathcal{A}_1\,(\{(q,r\circ t),(t,q\circ r)\})$	0.7421
$\mathcal{A}_1\,(\{(r,q\circ t),(t,q\circ r)\})$	**0.7447**
$\mathcal{A}_1\,(\{(r,q\circ t),(q,r\circ t)\})$	0.7435
$\mathcal{A}_1\,(\mathcal{P}_1)$	0.7303
$\mathcal{A}_2\,((r,q\circ t),(t,q\circ r))$	**0.7472**

Table 4: AVA F1 on AVA-WQA test set, using train and dev. sets from AVA-WQA.

- $\mathcal{A}_0\,(\{(q,r)\})$ as expected cannot predict if an answer is correct (its F1 is lower than 7%) since it does not use the answer representation.
- $\mathcal{A}_0\,(\{(q,t)\})$ is already a good model as it is as much powerful as a QA system.
- $\mathcal{A}_0\,(\{(r,t)\})$ is already a reasonable model, intuitively based on paraphrasing between r and t, but its F1 is 9% (62.47 vs 68.07) lower than $\mathcal{A}_0\,(\mathcal{P}_0)$, which uses all information, indicating that the semantic bias of q is essential to learn the right similarity between r and t.
- The results of the $\mathcal{A}_1$ models using a single triplet of q, r and t (i.e., 70.14, 73.87, 72.36) indicate that a text concatenation as input to Transformer models captures more information than concatenating the three separate embedding pairs, e.g., $\mathcal{A}_0\,(\{(r,t)\})$ only obtains 68.07. Interestingly, q text must be concatenated with t or r, to generate more effective features (2 or 4 points more).
- The triplet combination, e.g., $\mathcal{A}_1\big(\{r, q\circ t),(t, q\circ r)\}\big)$, provides an even more accurate model, while the redundant information from $\mathcal{A}_1\,(\mathcal{P}_1)$ does not produce benefits.
- Finally, the *Peer Attention* model applied to the best representations, e.g., $\mathcal{A}_1\big(\{r, q\circ t),(t, q\circ r)\}\big)$, boost them even more, reaching ∼75%. This is an important result, considering that the annotator agreement (the refer-

	Metrics	RMSE $\pm$ σ	Kendall τ	Kendall p
TREC-QA-Dev	P@1	0.000 $\pm$ 0.000	1.000	0.003
	MAP	0.040 $\pm$ 0.019	1.000	0.003
	MRR	0.015 $\pm$ 0.011	0.866	0.017
TREC-QA-Test	P@1	0.034 $\pm$ 0.018	1.000	0.003
	MAP	0.041 $\pm$ 0.029	0.867	0.017
	MRR	0.020 $\pm$ 0.012	1.000	0.003
WikiQA-Dev	P@1	0.000 $\pm$ 0.000	1.000	0.009
	MAP	0.050 $\pm$ 0.039	0.733	0.056
	MRR	0.063 $\pm$ 0.052	0.690	0.056
WikiQA-Test	P@1	0.079 $\pm$ 0.030	0.889	0.017
	MAP	0.081 $\pm$ 0.040	0.733	0.056
	MRR	0.095 $\pm$ 0.035	0.867	0.017

Table 5: System-wise evaluation on TREC-QA and WikiQA using AVA model, $\mathcal{A}_2\left((r, q \circ t), (t, q \circ r)\right)$.

ence is not available to them) is lower than 25%.

6.4 Results on system-wise evaluation

We evaluate the ability of AVA in predicting the Accuracy of QA systems as well as the performance of AS2 tasks. We conduct two evaluation studies with two public datasets, TREC-QA and WikiQA, and our SD dataset.

Results on public datasets For TREC-QA and WikiQA, we evaluated a bag of different models on the development and test sets and compared the results to the performance measured by AVA using one of the best models according to the point-wise evaluation, i.e., $\mathcal{A}_2\left((r, q \circ t), (t, q \circ r)\right)$.

More specifically, we apply each model m to select the best answer t from the list of candidates for q in the dataset. We first compute the performance of model m based on the provided annotations. The metrics include Accuracy or Precision-at-1 (P@1), MAP, and MRR.

We then run AVA for (q, t) using the GS answers of q as references, r. When multiple references are available, the final score of (q, t) is the average of AVA scores applied to different r. Before computing the Accuracy on the test set, we tune the AVA threshold to minimize the RMSE between the Accuracy (P@1) measured by AVA and GS, on the dev. set of each dataset. We use these thresholds to evaluate the results also on the test sets.

We considered six different systems built with one Compare-Aggregate (CNN) trained model and five other Transformers-based models. Four of the latter are collected from public resources[4] (Garg et al., 2020). These models differ in the architectures, BERT vs RoBERTa vs TANDA, and their training data; thus, their output is rather different. We removed questions that have no correct or no incorrect answers.

Table 5 reports the overall results averaged over the six models. We note that (i) if we set the threshold on the dev. set, the error on P@1 on the dev. set is 0, which should not surprise the reader as we fit such set. (ii) This is not the case for MAP, which is a much harder value to predict as it requires to estimate an entire ranking. (iii) On the TREC-QA test set, AVA has an error ranging from 2 to 4.1 points on any measure. (iv) On the WikiQA test set, the error is higher, reaching 9.5%, probably due to the fact that WikiQA data is rather different (more than TREC-QA data) from the data used for training AVA. (v) the Std. Dev. is low, suggesting that AVA can be used to estimate system performance, with an error ranging from 4% to 16.5% at 95% confidence, depending on measure and dataset.

Additionally, we compute the Kendall's Tau-b correlation between the ranking of the six systems sorted in order of performance (P@1) according to GS and AVA. We observe a perfect correlation on TREC-QA and a high correlation on WikiQA. This means that AVA can be used to determine if a model is better than another, which is desirable when developing and/or deploying new systems; the low p-values indicate reliable results.

Finally, Table 7 compares the performance evaluated with GS and AVA for all six models. It is interesting to note the high variability of the performance of our tested QA systems, e.g., P@1 ranges from 59.6 to 96.2 (with several intermediate results) on TREC-QA. Nevertheless, as shown in Table 5, the predictions of AVA are close to those from humans.

Results on SD We use the SD dataset in this evaluation to have a further system-wise evaluation. This differs from the one before as the systems' configurations and the data reflect an industrial scenario. The task is more challenging as the output is not just from one neural model, it comes from a combination of modules, ranging from query understanding, retrieval engine setting, indexed data, document and sentence filters, and finally, the adopted AS2 model. Additionally, the questions set is rather different from the one used for training. Table 6 reports the Accuracy of eight QA systems (S1, ..., S8) on the dev. and test sets, evaluated according to

[4] github.com/alexa/wqa_tanda

ADS Split	Evaluator	S1	S2	S3	S4	S5	S6	S7	S8	RMSE $\pm$ σ	Kendall τ	Kendall p
Dev (20%)	AVA	0.215	0.278	0.22	0.369	0.285	0.294	0.283	0.355	0.0198 $\pm$ 0.012	0.929	0.0004
	GS	0.218	0.282	0.234	0.379	0.309	0.315	0.261	0.319			
Test (80%)	AVA	0.235	0.289	0.235	0.355	0.319	0.321	0.301	0.357	0.0350 $\pm$ 0.019	0.643	0.031
	GS	0.235	0.324	0.26	0.393	0.356	0.365	0.249	0.336			

Table 6: Systems' P@1 evaluated with AVA and the GS annotations of SD

		Metrics	M1	M2	M3	M4	M5	M6
TREC-Dev	Gold	P@1	0.717	0.870	0.891	0.935	0.739	0.826
		MAP	0.691	0.858	0.913	0.912	0.769	0.796
		MRR	0.819	0.923	0.937	0.967	0.835	0.890
	AVA	P@1	0.717	0.870	0.891	0.935	0.739	0.826
		MAP	0.688	0.831	0.864	0.857	0.717	0.772
		MRR	0.809	0.920	0.940	0.967	0.803	0.876
TREC-Test	Gold	P@1	0.596	0.885	0.904	0.962	0.712	0.788
		MAP	0.661	0.873	0.894	0.904	0.771	0.801
		MRR	0.763	0.933	0.945	0.976	0.820	0.869
	AVA	P@1	0.635	0.904	0.962	0.981	0.712	0.827
		MAP	0.639	0.845	0.896	0.886	0.680	0.789
		MRR	0.764	0.936	0.981	0.990	0.793	0.880
WikiQA-Dev	Gold	P@1	0.545	0.727	0.455	0.545	0.636	0.727
		MAP	0.636	0.744	0.656	0.621	0.755	0.781
		MRR	0.720	0.831	0.695	0.703	0.803	0.864
	AVA	P@1	0.545	0.727	0.455	0.545	0.636	0.727
		MAP	0.523	0.751	0.643	0.617	0.713	0.774
		MRR	0.568	0.841	0.682	0.698	0.788	0.841
WikiQA-Test	Gold	P@1	0.563	0.844	0.781	0.688	0.813	0.781
		MAP	0.634	0.778	0.753	0.746	0.834	0.820
		MRR	0.746	0.917	0.876	0.833	0.906	0.883
	AVA	P@1	0.625	0.781	0.719	0.656	0.719	0.656
		MAP	0.660	0.750	0.687	0.683	0.705	0.704
		MRR	0.732	0.820	0.783	0.741	0.791	0.762

Table 7: Details of system-wise evaluation on TREC-QA and WikiQA using AVA model and GS, $\mathcal{A}_2\left((r, q \circ t), (t, q \circ r)\right)$

GS and AVA, along with RMSE and Kendall statistics of the two different evaluations. The RMSE is rather low 3.5% with a standard deviation of 1.9%, which indicates a max prediction error less than $\pm$7% with a confidence of 95%. The rank correlation is lower than what was observed on the academic benchmarks as the 8 evaluated systems have very close Accuracy. In any case, AVA can still be effectively used to select the top 3-4 systems.

6.5 Qualitative Analysis

Table 8 reports some example questions from TREC-QA test set, the top candidate selected by the TANDA system (Garg et al., 2020), the classification score of the latter, and the AVA score, which will determine a correct answer when it is larger than 0.5. For the first three questions, we note that, even thought the score of TANDA system is low, e.g., 0.0001, AVA can assign a rather high score, e.g., 0.596. In the first question, this is possible since AVA can match the winner of the literature prize, *Sully Prudhomme*, as well as the year of the event with the answer candidate. This match can not happen with the question.

In the second question, *Eileen Marie* can be matched with the question but there is basically no direct match between *branch of the service* and *to command a space shuttle mission* as *air force col*. In contrast, the reference provides easy matching, such as *air force colonel* and *command a space mission*. A similar rationale applies to the third question.

Conversely, a wrong answer could be classified as such by AVA, even if TANDA assigned it a very large score. For example, *1988* can be a reasonable date in an answer to the fourth question. This match prevents the selector to discard the answer. In contrast, the date above does not match with *1986* in the reference, and the importance of this mismatch is amplified by the presence of *when* in the question, which suggests AVA to pay attention to dates (in line with peer-attention modeling).

AVA vs. Overfitted reranker We investigated the performance of AVA in an open-domain setting, where the candidate answers are all sentences contained in the retrieved web documents.

Given a question, we analyzed the top-1 candidates reranked by two models: (i) a Transformer-based reranker fine-tuned on the same test questions (overfitting them); and (ii) the general AVA model using the answer the reranker was trained on, as reference. We used ASNQ (Garg et al., 2020) questions, which are typically associated with only one correct answer. For each question, we retrieved the top 200 relevant documents, $\sim$10,000 sentences, from a large index built with the 100MM documents from Common Crawl (`commoncrawl.org`), and used them as input of our models.

We manually evaluated the top-1 answer candidate produced by the reranker and AVA for 100 randomly selected questions. The results show that

Question q	Candidate t	TANDA	Reference r	$\mathcal{A}$
when were the nobel prize awards first given ?	among them is the winner of the first prize in 1901 , sully prudhomme .	0.0001 (correct)	leo tolstoy lost the first literature prize in 1901 to the forgettable rene f . a . sully prudhomme .	0.596
what branch of the service did eileen marie collins serve in ?	the first woman to command a space shuttle mission , air force col . eileen collins , sees her flight next month as `` a great challenge '' in more ways than one .	0.046 (correct)	shuttle commander eileen collins , a working mother and air force colonel , was set to make history as the first woman to command a space mission .	0.895
what was johnny appleseed 's real name ?	appleseed , whose real name was john chapman , planted many trees in the early 1800s .	0.026 (correct)	whitmore said he was most fascinated with the story of john chapman , who is better known as johnny appleseed .	0.948
when was the challenger space shuttle disaster ?	sept . 29 , 1988 _ americans return to space aboard the shuttle discovery , after a 32-month absence in the wake of the challenger accident .	0.995 (incorrect)	challenger was lost on its 10th mission during a 1986 launch accident that killed seven crew members .	0.080

Table 8: Examples show AVA can detect the failures of the state-of-the-art model by Garg et al. (2020).

Ques. q	Reference r	Overfitted Reranker	$\mathcal{A}$
when did apple computer change to apple inc	On January 9 , 2007 , Apple Computer , Inc. shortened its name to simply Apple Inc .	On January 9th, 2007 "Apple Computers" was renamed "Apple Inc." to reflect the shift in focus towards costumers electronics.	In 2007, Apple Computer, Inc. changed their name to Apple, Inc.
how much gold is there in fort knox	As of November 2017, Fort Knox holdings are 4,582 metric tons (147.3 million oz. troy).	At over 15 million ounces of gold, the deposit is one of the world's largest, located in an area designated for mining.[5]	According to official records, Fort Knox holds 4,578 metric tons of gold bullion, or roughly 2.5% of the entire world's known gold supply.
what muscle in the upper body covers the upper chest	The pectoralis major is a thick , fan - shaped muscle , situated at the chest (anterior) of the human body .	The upper portion of your back is referred to as the thoracic spine, and it includes the trapezius, rhomboids, teres muscles, infraspinatus, and lats.	Chest presses focus on exactly that–the chest muscle, called the pectoralis major.

Table 9: Examples show AVA can identify correct answers sharing the semantics of the questions.

AVA is much more accurate than the overfitted reranker, 66% versus 25%.

Table 9 shows some questions q, with their references r, and the answers selected by the two models. We note that the overfitted reranker selects answers that either (i) highly overlap with the reference (first example), or (ii) are typically wrong when such continuous word overlapping is missing (second and third examples).

In contrast, AVA selects answers that are rather different from the reference, even though they share the same semantics in answering the question.

7 Conclusion

We have presented AVA, the first automatic evaluator method for QA systems. We created seven different datasets, classified into three different types, which we used to develop AVA. We released those based on public data and plan to release the others. Then, we proposed different Transformer-based models and a new peer attention approach to capture answer and reference similarity induced by the question semantics. Our extensive experimentation has shown the AVA effectiveness for different types of evaluation: point-wise and system-wise over Accuracy, MAP and MRR. The results suggest that AVA can estimate the measures above, with a max error of 7% at 95% of confidence.

AVA can also be applied to generate distant supervision data. An example of this future application is given by (Krishnamurthy et al., 2021).

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL 2019*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Siddhant Garg, Thuy Vu, and Alessandro Moschitti.

2020. TANDA: transfer and adapt pre-trained transformer models for answer sentence selection. In *AAAI 2020*, pages 7780–7788. AAAI Press.

Sarik Ghazarian, Johnny Wei, Aram Galstyan, and Nanyun Peng. 2019. Better automatic evaluation of open-domain dialogue systems with contextualized embeddings. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 82–89, Minneapolis, Minnesota. Association for Computational Linguistics.

Tilani Gunawardena, Nishara Pathirana, Medhavi Lokuhetti, Roshan G. Ragel, and Sampath Deegalla. 2015. Performance evaluation techniques for an automatic question answering system. *International Journal of Machine Learning and Computing, Vol. 5, No. 4, August 2015*.

Anjuli Kannan and Oriol Vinyals. 2017. Adversarial evaluation of dialogue models. *CoRR*, abs/1701.08198.

Vivek Krishnamurthy, Thuy Vu, and Alessandro Moschitti. 2021. Reference-based weak supervision for answer sentence selection using web data.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *ICLR 2020*.

Jochen L. Leidner and Chris Callison-Burch. 2003. Evaluating question answering systems using faq answer injection. In *Proceedings of the 6th Annual CLUK Research Colloquium*.

Jimmy J. Lin and Dina Demner-Fushman. 2006. Methods for automatically evaluating answers to complex questions. *Information Retrieval*, 9:565–587.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic Turing test: Learning to evaluate dialogue responses. In *ACL 2017*, pages 1116–1126, Vancouver, Canada. Association for Computational Linguistics.

Qingsong Ma, Johnny Wei, Ondřej Bojar, and Yvette Graham. 2019. Results of the WMT19 metrics shared task: Segment-level and strong MT systems pose big challenges. In *WMT 2019*, pages 62–90, Florence, Italy. Association for Computational Linguistics.

Bernardo Magnini, Matteo Negri, Roberto Prevete, and Hristo Tanev. 2002. Towards automatic evaluation of question/answering systems. In *LREC 2002*.

Tom Mitchell, Terry Russell, Peter Broomhead, and Nicola Aldridge. 2002. Towards robust computerised marking of free-text responses.

Michael Mohler, Razvan Bunescu, and Rada Mihalcea. 2011. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *ACL 2011*, pages 752–762, Portland, Oregon, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL 2002*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Stephen G. Pulman and Jana Z. Sukkarieh. 2005. Automatic short answer marking. In *Proceedings of the Second Workshop on Building Educational Applications Using NLP*, pages 9–16, Ann Arbor, Michigan. Association for Computational Linguistics.

Brian Riordan, Andrea Horbach, Aoife Cahill, Torsten Zesch, and Chong Min Lee. 2017. Investigating neural architectures for short answer scoring. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 159–168, Copenhagen, Denmark. Association for Computational Linguistics.

Chirag Shah and Jefferey Pomerantz. 2010. Evaluating and predicting answer quality in community qa. In *SIGIR 2010*, pages 411–418, New York, NY, USA. ACM.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *EMNLP 2017*, pages 1179–1189, Copenhagen, Denmark. Association for Computational Linguistics.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2017. RUBER: an unsupervised method for automatic evaluation of open-domain dialog systems. *CoRR*, abs/1701.03079.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

E. Voorhees and D. Tice. 1999. *The TREC-8 Question Answering Track Evaluation*, pages 77–82. Department of Commerce, National Institute of Standards and Technology.

Tianqi Wang, Naoya Inoue, Hiroki Ouchi, Tomoya Mizumoto, and Kentaro Inui. 2019. Inject rubrics into short answer grading system. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for*

Low-Resource NLP (DeepLo 2019), pages 175–182, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP 2020: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Seunghyun Yoon, Franck Dernoncourt, Doo Soon Kim, Trung Bui, and Kyomin Jung. 2019. A compare-aggregate model with latent clustering for answer selection. In *CIKM 2019*, pages 2093–2096. ACM.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *ICLR 2020*.