QUASE: Question-Answer Driven Sentence Encoding

Hangfeng He[†]   **Qiang Ning**[‡*]   **Dan Roth**[†]
† University of Pennsylvania   ‡ Allen Institute for AI
`hangfeng,danroth@seas.upenn.edu`
`qiangn@allenai.org`

Abstract

Question-answering (QA) data often encodes essential information in many facets. This paper studies a natural question: Can we get supervision from QA data for other tasks (typically, non-QA ones)? For example, *can we use QAMR (Michael et al., 2017) to improve named entity recognition?* We suggest that simply further pre-training BERT is often not the best option, and propose the *question-answer driven sentence encoding (*QUASE*)* framework. QUASE learns representations from QA data, using BERT or other state-of-the-art contextual language models. In particular, we observe the need to distinguish between two types of sentence encodings, depending on whether the target task is a single- or multi-sentence input; in both cases, the resulting encoding is shown to be an easy-to-use plugin for many downstream tasks. This work may point out an alternative way to supervise NLP tasks.[1]

1 Introduction

It is labor-intensive to acquire human annotations for NLP tasks which require research expertise. For instance, one needs to know thousands of semantic frames in order to provide semantic role labelings (SRL) (Palmer et al., 2010). It is thus an important research direction to investigate how to get supervision signals from *indirect* data and improve one's target task. This paper studies the case of learning from question-answering (QA) data for other tasks (typically not QA). We choose QA because (1) a growing interest of QA has led to many large-scale QA datasets available to the community; (2) a QA task often requires comprehensive understanding of language and may encode rich information that

is useful for other tasks; (3) it is much easier to answer questions relative to a sentence than to annotate linguistics phenomena in it, making this a plausible supervision signal (Roth, 2017).

There has been work showing that QA data for task $\mathcal{A}$ can help another QA task $\mathcal{T}$, conceptually by further pre-training the *same* model on $\mathcal{A}$ (an often larger) before training on $\mathcal{T}$ (a smaller) (Talmor and Berant, 2019; Sun et al., 2019). However, it remains unclear how to use these QA data when the target task does not share the same model as the QA task, which is often the case when the target task is *not* QA. For instance, QA-SRL (He et al., 2015), which uses QA pairs to represent those predicate-argument structures in SRL, should be intuitively helpful for SRL parsing, but the significant difference in their surface forms prevents us from using the *same* model in both tasks.

The success of modern language modeling techniques, e.g., ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), and many others, has pointed out an alternative solution to this problem. That is, to further pre-train[2] a neural language model (LM) on these QA data *in certain ways*, obtain a sentence encoder, and use the sentence encoder for the target task, either by fine-tuning or as additional feature vectors. We call this general framework *question-answer driven sentence encoding (*QUASE*)*. A straightforward implementation of QUASE is to first further pre-train BERT (or other LMs) on the QA data in the standard way, as if this QA task is the target, and then fine-tune it on the real target task. This implementation is technically similar to STILTS (Phang et al., 2018), except that

---

[1]Our code and online demo are publicly available at `https://github.com/CogComp/QuASE`.

[2]We clarify three types of training: pre-training, further pre-training, and fine-tuning. Pre-training refers to the training of sentence encoders on unlabeled text; further pre-training refers to continuing training the sentence encoders on an intermediate, non target-task-specific labeled data (e.g. QA data); fine-tuning refers to training on the target task in the fine-tuning approach.

STILTS is mainly further pre-trained on textual entailment (TE) data.

However, similar to the observations made in STILTS and their follow-up works (Wang et al., 2019), we find that additional QA data does not necessarily help the target task using the implementation above. While it is unclear how to predict this behaviour , we do find that this happens a lot for tasks whose input is a single sentence, e.g., SRL and named entity recognition (NER), instead of a sentence pair, e.g., TE. This might be because QA is itself a paired-sentence task, and the implementation above (i.e., to further pre-train BERT on QA data) may learn certain attention patterns that can transfer to another paired-sentence task more easily than to a single-sentence task. Therefore, we argue that, for single-sentence target tasks, QUASE should restrict the interaction between the two sentence inputs when it further pre-trains on QA data. We propose a new neural structure for this and name the resulting implementation s-QUASE, where "s" stands for "single;" in contrast, we name the straightforward implementation mentioned above p-QUASE for "paired." Results show that s-QUASE outperforms p-QUASE significantly on 3 single-sentence tasks—SRL, NER, and semantic dependency parsing (SDP)—indicating the importance of this distinction.

Let $\text{QUASE}_{\mathcal{A}}$ be the QUASE further pre-trained on QA data $\mathcal{A}$. We extensively compare 6 different choices of $\mathcal{A}$: TriviaQA (Joshi et al., 2017), NewsQA (Trischler et al., 2017), SQuAD (Rajpurkar et al., 2016), relation extraction (RE) dataset in QA format (QA-RE for short) (Levy et al., 2017), Large QA-SRL (FitzGerald et al., 2018), and QAMR (Michael et al., 2017). Interestingly, we find that if we use s-QUASE for single-sentence tasks and p-QUASE for paired-sentence tasks, then $\text{QUASE}_{\text{QAMR}}$ improves all 7 tasks[3] in low resource settings, with an average error reduction rate of 7.1% compared to BERT.[4] While the set of tasks we experimented with here is non-exhaustive, we think that $\text{QUASE}_{\text{QAMR}}$ has the potential of improving on a wide range of tasks.

This work has three important implications. First, it provides supporting evidence to an important alternative to supervising NLP tasks: using QA to annotate language, which has been discussed in works such as QA-SRL, QAMR, and QA-RE. If it is difficult to teach annotators the formalism of a certain task, perhaps we can instead collect QA data that query the target phenomena and thus get supervision from QA for the original task (and possibly more). Second, the distinction between s-QUASE and p-QUASE suggests that sentence encoders should consider some properties of the target task (e.g., this work distinguishes between single- and multi-sentence tasks). Third, the good performance of $\text{QUASE}_{\text{QAMR}}$ suggests that predicate-argument identification is an important capability that many tasks rely on; in contrast, many prior works observed that only language modeling would improve target tasks generally.

2 QA Driven Sentence Encoding

This work aims to find an effective way to use readily available QA data to improve a target task that is typically not QA. A natural choice nowadays—given the success of language models—is to further pre-train sentence encoders, e.g. BERT, on QA data *in certain ways*, and then use the new encoder in a target task. This general framework is called QUASE in this work, and the assumption is that the sentence encoders learned from QA data have useful information for the target task.

A straightforward implementation of QUASE is to further pre-train BERT on QA data *in the standard way*, i.e., fine-tune BERT as if this QA dataset is the target task, and then fine-tune BERT on the real target task. However, we find that this straightforward implementation is less effective or even negatively impacts target tasks with single-sentence input; similar observations were also made in STILTS (Phang et al., 2018) and its follow-ups (Wang et al., 2019): They further pre-train sentence encoders, e.g., ELMo, BERT, and GPT (Radford et al., 2018), on TE data and find that it is not effective for the syntax-oriented CoLA task and the SST sentiment task in GLUE, which are both single-sentence tasks (Wang et al., 2018).

One plausible reason is that the step of further pre-training on QA data does not take into account some properties of the target task, for instance, the number of input sentences. QA is inherently a paired-sentence task; a typical setup is, given a context sentence and a question sentence, predict the answer span. Further pre-training BERT on QA data will inevitably learn how to attend to the context given the question. This is preferable when the target task is also taking a pair of sentences

---

[3]SRL, SDP, NER, RE, co-reference resolution (Coref), TE and machine reading comprehension (MRC).

[4]BERT is close to the state-of-the-art in all these tasks.

(a) s-QuASE for single-sentence tasks
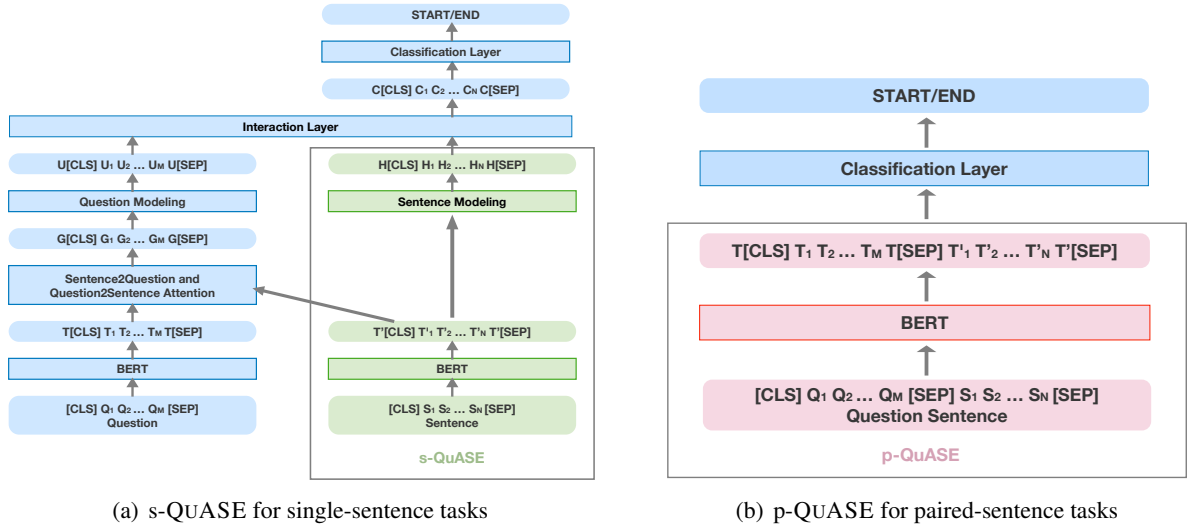
(b) p-QuASE for paired-sentence tasks

Figure 1: Two implementations of QuASE: s-QuASE for single-sentence tasks, and p-QuASE for paired-sentence tasks. Both structures are further pre-trained on QA data, and the parts in the black boxes are used by target tasks. While p-QuASE is the standard way of fine-tuning BERT , s-QuASE restricts the interaction between the sentence and the question. Specifically, the sentence encodings in s-QuASE does not depend on the existence of the question. More details are given in Sec. 2.2 and Appendix A (including experimental settings in A.1, error analysis in A.2 and ablation analysis in A.3).

as input, while it may be irrelevant or harmful for single-sentence tasks. It points out that we may need two types of sentence encodings when further pre-training BERT on QA data, depending on the type of the target task. The following subsection discusses this issue in detail.

2.1 Two Types of Sentence Encodings

Standard sentence encoding is the problem of converting a sentence $S=[w_1, w_2, \cdots, w_n]$ to a sequence of vectors $h(S)=[h_1, h_2, \cdots, h_n]$ (e.g., skip-thoughts (Kiros et al., 2015)). Ideally, $h(S)$ should encode *all* the information in S, so that it is task-agnostic: given a target task, one can simply probe $h(S)$ and retrieve relevant information. In practice, however, only the information relevant to the training task of $h(S)$ is kept. For instance, when we have a task with multi-sentence input (e.g., QA and TE), the attention pattern A among these sentences will affect the final sentence encoding, which we call $h_A(S)$; in comparison, we denote the sentence encoding learned from single-sentence tasks by $h(S)$, since there is no cross-sentence attention A. In a perfect world, the standard sentence encoding $h(S)$ expresses also the conditional sentence encoding $h_A(S)$. However, we believe that there is a trade-off between the quality and the quantity of semantic information a model can encode. Our empirical results corroborate this conclusion and

more details can be found in Appendix A.2.

The distinction between the sentence encodings types may explain the negative impact of using QA data for some single-sentence tasks: Further pre-training BERT on QA data essentially produces a sentence encoding with cross-sentence attentions $h_A(S)$, while the single-sentence tasks expect $h(S)$. These two sentence encodings may be very different: One view is from the theory of information bottleneck (Tishby et al., 1999; Tishby and Zaslavsky, 2015), which argues that training a neural network on a certain task is extracting an approximate *minimal* sufficient statistic of the input sentences with regard to the target task; information irrelevant to the target task is *maximally* compressed. In our case, this corresponds to the process where the conditional sentence encoding compresses the information irrelevant to the relation, which will enhance the quality but reduce the quantity of the sentence information.

2.2 Two Implementations of QuASE

In order to fix this issue, we need to know how to learn $h(S)$ from QA data. However, since QA is a paired-sentence task, the attention pattern between the context sentence and the question sentence is important for successful further pre-training on QA. Therefore, we propose that if the target task is single-sentence input, then fur-

8745

ther pre-training on QA data should also focus on single-sentence encodings in the initial layers; the context sentence should not interact with the question sentence until the very last few layers. This change is expected to hurt the capability to solve the auxiliary QA task, but it is later proved to transfer better to the target task. This new treatment is called s-QUASE with "s" representing "single-sentence," while the straightforward implementation mentioned above is called p-QUASE where "p" means "paired-sentence." The specific structures are shown in Fig. 1.

2.2.1 s-QUASE

The architecture of s-QUASE is shown in Fig. 1(a). When further pre-training it on QA data, the context sentence and the question sentence are fed into two pipelines. We use the same Sentence2Question and Question2Sentence attention as used in BiDAF (Seo et al., 2017). Above that, "Sentence Modeling," "Question Modeling," and "Interaction Layer" are all bidirectional transformers (Vaswani et al., 2017) with 2 layers, 2 layers, and 1 layer, respectively. Finally, we use the same classification layer as BERT, which is needed for training on QA data. Overall, this implementation restricts interactions between the paired-sentence input, especially from the question to the context, because when serving the target task, this attention will not be available.

Using s-QUASE in target tasks. Given a sentence S, s-QUASE can provide a sequence of hidden vectors $h(S)$, i.e., the output of the "Sentence Modeling" layer in Fig. 1(a). Although $h(S)$ does not rely on the question sentence, $h(S)$ is optimized so that upper layers can use it to handle those questions in the QA training data, so $h(S)$ indeed captures information related to the phenomena queried by those QA pairs. For single-sentence tasks, we use $h(S)$ from s-QUASE as additional features, and concatenate it to the word embeddings in the input layer of any specific neural model.[5]

2.2.2 p-QUASE

The architecture of p-QUASE is shown in Fig. 1(b), which is the standard way of pre-training BERT. That is, when further pre-training it on QA data, the context sentence and the question sentence form a single sequence (separated by special tokens) and are fed into BERT.

---

[5]We mainly use concatenation in both types of QUASE. However, we also use replacement in some experiments and we will note these cases later in this paper.

Using p-QUASE in target tasks. Given a sentence pair S (concatenated), p-QUASE produces $h_A(S)$, i.e., the output of the BERT module in Fig. 1(b). One can of course continue fine-tuning p-QUASE on the target task, but we find that adding p-QUASE to an existing model for the target task is empirically better (although not very significant); specifically, we try to add $h_A(S)$ to the final layer before the classification layer, and we also allow p-QUASE to be updated when training on the target task, although it is conceivable that other usages may lead to even stronger results. For instance, when the target task is token classification, e.g., MRC, we can simply concatenate the vectors of $h_A(S)$ at each timestamp to any existing model; when the target task is sentence classification, e.g., TE, we apply max-pooling and average-pooling on $h_A(S)$, respectively, and concatenate the two resulting vectors to any existing model before the final classification layer.

2.3 Related Work on Sentence Encoding

Modern LMs are essentially sentence encoders pre-trained on unlabeled data and they outperform early sentence encoders such as skip-thoughts (Kiros et al., 2015). While an LM like BERT can handle lexical and syntactic variations quite well, it still needs to learn from some annotations to acquire the "definition" of many tasks, especially those requiring complex semantics (Tenney et al., 2019). Although we extensively use BERT here, we think that the specific choice of LM is orthogonal to our proposal of learning from QA data. Stronger LMs, e.g., RoBERTa (Liu et al., 2019) or XLNet (Yang et al., 2019), may only strengthen the proposal here. This is because a stronger LM represents *unlabeled* data better, while the proposed work is about how to represent *labeled* data better.

CoVe (McCann et al., 2017) is another attempt to learn from indirect data, translation data specifically. However, it does not outperform ELMo or BERT in many NLP tasks (Peters et al., 2018) and probing analysis (Tenney et al., 2019). In contrast, our QUASE will show stronger experimental results than BERT on multiple tasks. In addition, we think QA data is generally cheaper to collect than translation data.

The proposed work is highly relevant to Phang et al. (2018) and their follow-up works (Wang et al., 2019), which use further pre-training on data-rich intermediate supervised tasks and aim

System	Single-sentence		Paired-sentence	
	SRL	RE	TE	MRC
BERT	**34.17**	**62.99**	78.29	79.90
BERT_{QAMR}	32.92	50.16	**78.73**	**82.96**

Table 1: The naive way of training BERT on QAMR (BERT_{QAMR}) negatively impacts single-sentence tasks. We only use 10% training data for simplicity. We use BERT/BERT_{QAMR} to produce feature vectors for a BiLSTM model (SRL) and a CNN model (RE); for TE and MRC, we fine-tune BERT/BERT_{QAMR}.

to improve another target task. The key differences are as follows: First, we distinguish two types of sentence encodings, which provide explanation to their puzzle that sentence-pair tasks seem to benefit more from further pre-training than single-sentence tasks do. Second, they only focus on fine-tuning based methods which cannot be easily plugged in many single-sentence tasks such as SRL and Coref, while we analyze both fine-tuning based and feature-based approaches. Third, they mainly use TE signals for further pre-training, and evaluate their models on GLUE (Wang et al., 2018) which is a suite of tasks very similar to TE. Our work instead makes use of QA data to help tasks that are typically not QA. Fourth, from their suite of further pre-training tasks, they observe that only further pre-training on language modeling tasks has the power to improve a target task in general, while we find that QAMR may also have this potential, indicating the universality of predicate-argument structures in NLP tasks.

Our work is also related to Sentence-BERT (Reimers and Gurevych, 2019) in terms of providing a better sentence representation. However, their focus was deriving semantically meaningful sentence embeddings that can be compared using cosine-similarity, which reduces the computational cost of finding the most similar pairs. In contrast, QuASE provides a better sentence encoder in the same format as BERT (a sequence of word embeddings) to better support tasks that require complex semantics.

3 Applications of QuASE

In this section, we conduct thorough experiments to show that QuASE is a good framework to get supervision from QA data for other tasks. We first give an overview of the datasets and models used in these experiments before diving into the details

of each experiment.

Specifically, we use PropBank (Kingsbury and Palmer, 2002) (SRL), the dataset from the SemEval'15 shared task (Oepen et al., 2015) with DELPH-IN MRS-Derived Semantic Dependencies target representation (SDP), CoNLL'03 (Tjong Kim Sang and De Meulder, 2003) (NER), the dataset in SemEval'10 Task 8 (Hendrickx et al., 2009) (RE), the dataset in the CoNLL'12 shared task (Pradhan et al., 2012) (Coref), MNLI (Williams et al., 2018) (TE), and SQuAD 1.0 (Rajpurkar et al., 2016) (MRC). In Table 4, we use CoNLL'12 English subset of OntoNotes 5.0 (Pradhan et al., 2013), which is larger than PropBank. The performance of TE and MRC is evaluated on the development set.[6]

For single-sentence tasks, we use both simple baselines (e.g., BiLSTM and CNN; see Appendix B.1) and near-state-of-the-art models published in recent years. As in ELMo, we use the deep neural model in He et al. (2017) for SRL, the model in Peters et al. (2018) for NER, and the end-to-end neural model in Lee et al. (2017) for Coref. We also use the biaffine network in Dozat and Manning (2018) for SDP but we removed part-of-speech tags from its input, and the attention-based BiLSTM in Zhou et al. (2016) is the strong baseline for RE. In addition, we replace the original word embeddings in these models (e.g., GloVe (Pennington et al., 2014)) by BERT. Throughout this paper, we use the pre-trained case-insensitive BERT-base implementation. More details on our experimental setting can be found in Appendix B, including the details of simple models in B.1, some common experimental settings of QuASE in B.2, and s-QuASE combined with other SOTA embeddings (ELMo and Flair (Akbik et al., 2018)) in B.3.

3.1 Necessity of Two Representations

We first consider a straightforward method to use QA data for other tasks—to further pre-train BERT on these QA data. We compare BERT further pre-trained on QAMR (denoted by BERT_{QAMR}) with BERT on two single-sentence tasks (SRL and RE) and two paired-sentence tasks (TE and MRC). We use a feature-based approach for single-sentence tasks and a fine-tuning approach for paired-sentence tasks. The reason is two-fold. On the one hand, current SOTAs of all single-sentence tasks considered in this paper are still

---

[6]For TE, we mean matched examples in MNLI.

	Single-Sentence Tasks						Paired-Sentence Tasks			
Tasks	SRL		SDP		NER		TE		MRC	
Split	10%	100%	10%	100%	10%	100%	10%	30%	10%	100%
s-QUASE	**46.42**	**70.13**	**76.08**	**87.29**	**70.69**	**87.10**	52.25	57.30	44.67	67.09
p-QUASE	32.92	66.40	70.92	86.43	49.97	85.23	**57.29**	**60.49**	**48.29**	**72.97**

Table 2: Probing results of the sentence encoders from s-QUASE and p-QUASE. In all tasks, we fix the model QUASE and use the sentence encodings as input feature vectors for the model of each task. In order to keep the model structure as simple as possible, we use BiLSTM for SRL, NER, and TE, Biaffine for SDP, and BiDAF for MRC. We compare on 10% and 100% of the data in all tasks except TE, where we use 30% to save run-time.
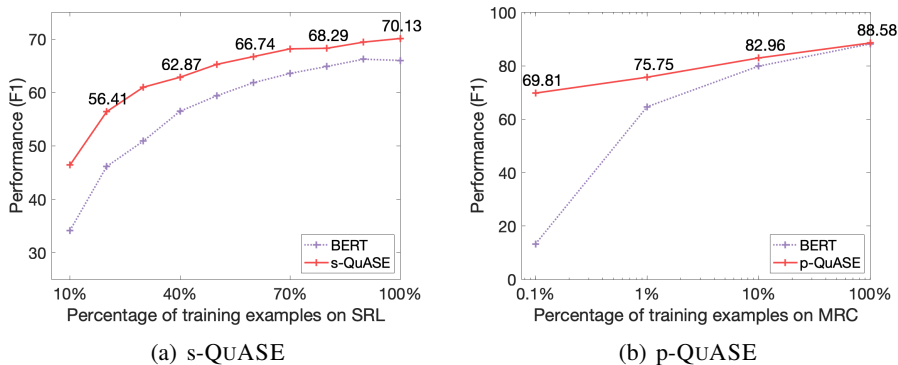

(a) s-QUASE

(b) p-QUASE

Figure 2: Sample complexity analysis of using BERT and QUASE on SRL and MRC. We find that much fewer training examples are needed with the help of QUASE$_{QAMR}$: with 50% SRL training data, s-QUASE can achieve comparable performance as BERT trained on 100%; with 0.1% training data for MRC, p-QUASE can achieve a reasonably good performance of 69.81%.

feature-based. How to efficiently use sentence encoders (e.g. BERT) in a fine-tuning approach for some complicated tasks (e.g. SRL and SDP) is unclear. On the other hand, the fine-tuning approach shows great advantage over feature-based on many paired-sentence tasks (e.g. TE and MRC). Similar to Phang et al. (2018), we find in Table 1 that the two single-sentence tasks benefit less than the two paired-sentence tasks from BERT$_{QAMR}$, which indicates that simply "further pre-training BERT" is not enough.

We then compare s-QUASE$_{QAMR}$ and p-QUASE$_{QAMR}$ on three single-sentence tasks (SRL, SDP and NER) and two paired-sentence tasks (TE and MRC) to show that it is important to distinguish two types of sentence representations. Rather than concatenating two embeddings as proposed in Sec. 2.2, here we replace BERT embeddings with QUASE embeddings for convenience. The results are shown in Table 2. We find that s-QUASE has a great advantage over p-QUASE on single-sentence tasks and p-QUASE is better than s-QUASE on paired-sentence tasks. The proposal of two types of sentence encoders tackles the problem one may encounter when there is only further

pre-training BERT on QAMR for single-sentence tasks. In summary, it is necessary to distinguish two types of sentence representations for single-sentence tasks and paired-sentence tasks.

3.2 Sample Complexity of QUASE

To see whether adding QUASE to BERT reduces the sample complexity, we compare QUASE$_{QAMR}$ with BERT on one single-sentence task (SRL) and one paired-sentence task (MRC) with different percentages of training examples. For convenience, we replace BERT embeddings with QUASE embeddings for SRL. As shown in Figure 2, we find that s-QUASE$_{QAMR}$ outperforms BERT on SRL with small training data, and p-QUASE$_{QAMR}$ outperforms BERT on MRC with small training data. The results support that (1) adding QUASE to BERT reduces the sample complexity, (2) QUASE is very important in the low-resource setting. For instance, s-QUASE$_{QAMR}$ achieves an F1 score of 61 in SRL with 30% ($27K$) training examples (compared to 50.92 F1 by BERT). And p-QUASE$_{QAMR}$ achieves 69.81 average F1 on MRC with 0.1% (about 100) training examples (compared to 13.29 F1 by BERT).

Models	s-QUASE								p-QUASE			
Tasks	SRL		SDP		NER		RE		TE		Avg	
Split	small	full	small	full	small	full	small	full	small	full	small	full
BERT	34.17	66.02	75.49	90.13	88.89	91.38	71.48	86.33	78.29	84.09	69.66	83.59
QUASE	**50.16**	**72.59**	**78.30**	**90.78**	**90.64**	**92.16**	**77.14**	**86.80**	**78.94**	**84.97**	**75.04**	**85.46**
TriviaQA	17.75	39.69	77.29	90.43	89.74	91.70	75.41	**86.80**	78.50	84.95	67.74	78.71
NewsQA	27.99	53.05	77.27	90.41	89.96	91.65	73.08	85.88	78.85	84.30	69.43	81.06
SQuAD	34.35	61.86	76.90	90.51	89.94	91.07	**77.14**	85.80	78.21	**84.97**	71.31	82.84
QA-RE	35.50	65.85	**78.30**	**90.78**	**90.64**	91.73	63.36	85.80	**78.94**	84.68	69.35	83.77
Large QA-SRL	**50.16**	**72.59**	76.92	90.68	90.12	91.73	68.99	85.46	78.88	84.61	**73.01**	**85.01**
QAMR	46.42	70.13	77.53	90.57	89.90	**92.16**	72.23	86.37	78.73	84.79	72.96	84.80

Table 3: Further pre-training QUASE on different QA datasets of the same number of QA pairs ($51K$). As we propose, s-QUASE is used as features for single-sentence tasks, and p-QUASE is further fine-tuned for the paired-sentence task. The specific models are all strong baselines except for SRL, where we use a simple BiLSTM model to save run-time. "Small" means 10% training examples for all tasks except NER, where "small" means the dev set (about 23%) of the corresponding training set. We further show the results of QUASE with the best QA dataset, which are significantly better than those of BERT.

	Single-Sentence Tasks					Paired-Sentence Tasks		
Small	SRL	SDP	NER	RE	Coref	TE	MRC	Avg
BERT	76.65	75.49	88.89	71.48	62.76	78.29	79.90	76.21
Proposed (abs. imp.)	+3.95	+2.04	+1.01	+0.75	+0.60	+0.44	+3.06	+1.69
Proposed (rel. imp.)	16.9%	8.3%	9.1%	2.6%	1.6%	2.0%	15.2%	7.1%
Full	SRL	SDP	NER	RE	Coref	TE	MRC	Avg
BERT	84.54	90.13	91.38	86.33	69.05	84.09	88.23	84.82
Proposed (abs. imp.)	+0.15	+0.44	+0.78	+0.04	-0.14	+0.7	+0.35	+0.33
Proposed (rel. imp.)	0.9%	4.5%	9.0%	0.3%	-0.5%	4.4%	3.0%	2.2%

Table 4: QUASE$_{QAMR}$ (almost) universally improves on 5 single-sentence tasks and 2 paired-sentence tasks. Note BERT is close to the state of the art for these tasks. Both absolute improvement (abs. imp.) and relative improvement (rel. imp.; error reduction rate) are reported. "Small/Full" refers to the size of training data for each target task. For SDP, RE, TE, and MRC, "small" means 10% of the training set, while for NER, SRL, and Coref, "small" means the development set (about 10%-30% compared to each training set).

3.3 Data Choice for Further Pre-training

We compare BERT with QUASE further pre-trained with the same numbre of QA pairs on 6 different QA datasets (TriviaQA (Joshi et al., 2017), NewsQA (Trischler et al., 2017), SQuAD, QA-RE (Levy et al., 2017), Large QA-SRL (FitzGerald et al., 2018), and QAMR). s-QUASE further pre-trained on different QA datasets are evaluated on four single-sentence tasks in a feature-based approach: SRL, SDP, NER and RE. p-QUASE further pre-trained on different QA datasets is evaluated on one task (TE) in a fine-tuning approach.

In Table 3, we find that the best options are quite different across different target tasks, which is expected because a task usually benefits more from a more similar QA dataset. However, we also find that QAMR is generally a good further-pre-training choice for QUASE. This is consistent with our intuition: First, QAMR has a simpler concept class than other paragraph-level QA datasets, such as TriviaQA, NewsQA and SQuAD. It is easier for QUASE to learn a good representation with QAMR to help sentence-level tasks. Second, QAMR is more general than other sentence-level QA datasets, such as QA-RE and Large QA-SRL.[7] Therefore, we think that the capability to identify predicate-argument structures can generally help many sentence-level tasks, as we discuss next.

3.4 The Effectiveness of QUASE

Here we compare QUASE$_{QAMR}$ with BERT on 5 single-sentence tasks and 2 paired-sentence tasks, where QUASE$_{QAMR}$ is further pre-trained on the training set ($51K$ QA pairs) of the QAMR dataset. As shown in Table 4, we find that QUASE$_{QAMR}$

---

[7]Although the average performance of QUASE$_{QAMR}$ on five tasks is slightly below QUASE$_{Large\ QA-SRL}$, for which the benefit mostly comes from SRL. QUASE is mainly designed to improve a lot of tasks, so QAMR is a better choice in our setup, but in practice, we do not limit QUASE to any specific QA dataset and one can use the best one for corresponding target tasks.

has a better performance than BERT on both single-sentence tasks and paired-sentence tasks, especially in the low-resource setting[8], indicating that QUASE_{QAMR} can provide extra features compared to BERT.

Admittedly, the improvement in the "Full" setting is not significantly large, but we think that this is expected because large direct training data are available (such as SRL with $278K$ training examples in OntoNotes). However, it is still promising that $51K$ indirect QA pairs can improve downstream tasks in the low-resource setting (i.e. several thousands direct training examples). That is because they help the scalability of machine learning methods, especially for some specific domains or some low-resource languages where direct training data do not exist in large scale.

4 Discussion

In this section we discuss a few issues pertaining to improving QUASE by using additional QA datasets and the comparison of QUASE with related symbolic representations.

4.1 Further Pre-training QUASE on Multiple QA Datasets

We investigate whether adding the Large QA-SRL dataset (FitzGerald et al., 2018) or the QA-RE[9] dataset into QAMR in the further pre-training stage can help SRL and RE. We use s-QUASE embeddings to replace BERT embeddings instead of concatenating the two embeddings. The effectiveness of adding existing resources (Large QA-SRL or QA-RE) into QAMR in the further pre-training stage of s-QUASE on SRL and RE are shown in Table 5. We find that adding related QA signals (Large QA-SRL for SRL and QA-RE for RE) into QAMR can help improve specific tasks. Noteworthy is the fact that QA-RE can help SRL (Large QA-SRL can also help RE), though the improvement is minor compared to Large QA-SRL (QA-RE). These results indicate that adding more QA signals related to the sentence can help get a better sentence representation in general.

Tasks	SRL		RE	
Split	10%	100%	10%	100%
BERT	34.16	66.02	59.36	83.28
QUASE_{QAMR}	46.42	70.13	61.09	82.22
$\text{QUASE}_{QAMR+Large\ QA-SRL}$	**49.92**	71.74	65.76	83.16
$\text{QUASE}_{QAMR+QA-RE}$	47.25	**72.52**	**68.12**	**83.89**

Table 5: The potential of further improving QUASE_{QAMR} by further pre-training it on more QA data. The "+" between datasets means union with shuffling. Both Large QA-SRL and QA-RE help achieve better results than QAMR alone. For simplicity, we use a simple BiLSTM model for SRL and a simple CNN model for RE. See more in Appendix B.

4.2 Comparison with Symbolic Meaning Representations

Traditional (symbolic) shallow meaning representations such as SRL and AMR, suffer from having a fixed set of relations one has to commit to. Moreover, inducing these representations requires costly annotation by experts. Proposals such as QA-SRL, QAMR, semantic proto-roles (Reisinger et al., 2015), and universal dependencies (White et al., 2016) avoid some of these issues by using natural language annotations, but it is unclear how other tasks can take advantage of them. QUASE is proposed to facilitate inducing distributed representations instead of symbolic representations from QA signals; it benefits from cheaper annotation and flexibility, and can also be easily used in downstream tasks.

The following probing analysis, based on the Xinhua subset in the AMR dataset, shows that s-QUASE_{QAMR} embeddings encode more semantics related to AMR than BERT embeddings. Specifically, we use the same edge probing model as Tenney et al. (2019), and find that the probing accuracy (73.59) of s-QUASE_{QAMR} embeddings is higher than that (71.58) of BERT. At the same time, we find that p-QUASE_{QAMR} can achieve 76.91 F1 on the PTB set of QA-SRL, indicating that p-QUASE_{QAMR} can capture enough information related to SRL to have a good zero-shot SRL performance. More details can be found in Appendix C.1. Another fact worth noting is that AMR can be used to improve downstream tasks, such as MRC (Sachan and Xing, 2016), TE (Lien and Kouylekov, 2015), RE (Garg et al., 2019) and SRL (Song et al., 2018). The benefits of QUASE_{QAMR} on downstream tasks show that we can take advantage of AMR by learning from much cheaper QA signals dedicated to it.

---

[8]Another interesting finding is that simple models usually benefit more from QUASE embeddings than SOTA models.

[9]Because the training set of QA-RE is too large, we randomly choose $100,000$ training examples.

4.3 Difficulties in Learning Symbolic Representations from QA Signals

QUASE is designed to learn distributed representations from QA signals to help down-stream tasks. We further show the difficulties of learning two types of corresponding symbolic representations from QA signals, which indicates that the two other possible methods are not as tractable as ours.

One option of symbolic representation is the QAMR graph. Michael et al. (2017) show that question generation for QAMR representations can only achieve a precision of 28%, and a recall of 24%, even with fuzzy matching (multi-BLEU[10] > 0.8). Furthermore, it is still unclear how to use the complex QAMR graph in downstream tasks. *These results indicate that learning a QAMR parser for down-stream tasks is mainly hindered by question generation, and how to use the full information of QAMR for downstream tasks is still unclear.*

Another choice of symbolic representation is AMR, since QAMR is proposed to replace AMR. We consider a simpler setting, learning an SRL parser from Large QA-SRL. We propose three models in different perspectives, but the best performance of them is only 54.10 F1, even with fuzzy matching (Intersection/Union ≥ 0.5). More details can be found in Appendix C.2. Although a lot of methods (Khashabi et al., 2018; Marcheggiani and Titov, 2017; Strubell et al., 2018) can be adopted to use SRL/AMR in downstream tasks, *the difficulty of learning a good SRL/AMR parser from QA signals hinders this direction.*

The difficulties of learning the two types of symbolic representations from QA signals indicate that our proposal of learning distributed representations from QA signals is a better way of making use of the latent semantic information in QA pairs for down-stream tasks.

5 Conclusion

In this paper, we investigate an important problem in NLP: Can we make use of low-cost signals, such as QA data, to help related tasks? We retrieve signals from sentence-level QA pairs to help NLP tasks via two types of sentence encoding approaches. For tasks with a single-sentence input, such as SRL and NER, we propose s-QUASE that provides latent sentence-level representations; for tasks with a sentence pair input, such as TE and MRC we propose p-QUASE, that generates latent

---

[10]An average of BLEU1–BLEU4 scores.

representations related to attentions. Experiments on a wide range of tasks show that the distinction of s-QUASE and p-QUASE is highly effective, and QUASE$_{QAMR}$ has the potential to improve on many tasks, especially in the low-resource setting.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING*, pages 1638–1649.

M. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 280–287.

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

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *ACL*, pages 484–490.

Nicholas FitzGerald, Julian Michael, Luheng He, and Luke Zettlemoyer. 2018. Large-scale QA-SRL parsing. In *ACL*, pages 2051–2060.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform.

Sahil Garg, Aram Galstyan, Greg Ver Steeg, Irina Rish, Guillermo Cecchi, and Shuyang Gao. 2019. Kernelized hashcode representations for relation extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6431–6440.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *ACL*, pages 473–483.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language.

In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 643–653.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611. Association for Computational Linguistics.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2018. Question answering as global reasoning over semantic abstractions. In *Proceedings of The Conference on Artificial Intelligence (Proc. of the Conference on Artificial Intelligence (AAAI))*.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*, pages 1989–1993. Citeseer.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*, pages 188–197.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *CoNLL*, pages 333–342.

Elisabeth Lien and Milen Kouylekov. 2015. Semantic parsing for textual entailment. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 40–49.

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. *arXiv preprint arXiv:1907.11692*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *EMNLP*, pages 1506–1515.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NeurIPS*, pages 6294–6305.

Julian Michael, Gabriel Stanovsky, Luheng He, Ido Dagan, and Luke Zettlemoyer. 2017. Crowdsourcing question-answer meaning representations. *NAACL*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926.

M. Palmer, D. Gildea, and N. Xue. 2010. *Semantic Role Labeling*, volume 3.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*, pages 2227–2237.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *CoNLL*, pages 143–152.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3973–3983.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *Transactions of the Association for Computational Linguistics*, 3:475–488.

Dan Roth. 2017. Incidental supervision: Moving beyond supervised learning. In *Proc. of the Conference on Artificial Intelligence (AAAI)*.

Mrinmaya Sachan and Eric Xing. 2016. Machine comprehension using rich semantic representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 486–492.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *ICLR*.

Li Song, Yuan Wen, Sijia Ge, Bin Li, Junsheng Zhou, Weiguang Qu, and Nianwen Xue. 2018. An easier and efficient framework to annotate semantic roles: Evidence from the chinese amr corpus. In *The 13th Workshop on Asian Language Resources*, page 29.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *EMNLP*, pages 5027–5038.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019. Improving machine reading comprehension with general reading strategies. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2633–2643, Minneapolis, Minnesota. Association for Computational Linguistics.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4911–4921. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. *ICLR*.

Naftali Tishby, Fernando C Pereira, and William Bialek. 1999. The information bottleneck method. In *Proc. of the Annual Allerton Conference on Communication, Control and Computing*.

Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop (ITW)*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200. Association for Computational Linguistics.

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*, pages 5998–6008.

Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, et al. 2019. Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4465–4476.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal Decompositional Semantics on Universal Dependencies. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*, pages 1112–1122.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *ACL*, pages 207–212.

A Additional Details for QUASE

In this section, we show the experimental details of QUASE. We first show the experimental settings of training p-QUASE and s-QUASE in Section A.1. After that, we conduct error analysis of QUASE to show the shortcomings of QUASE in Section A.2. Finally, the ablation analysis of s-QUASE is in Section A.3.

A.1 Experimental Settings

Our QUASE is based on the re-implementation of BERT with pytorch (Wolf et al., 2019). Although we might change a bit to fit the memory of GPU sometimes, the common hyper parameters for further pre-training s-QUASE and p-QUASE are as follows:

Further pre-training p-QUASE. For sentence-level QA datasets (QAMR, Large QA-SRL, and QA-RE), we further pre-train BERT for 4 epochs with a learning rate of $5e$-5, a batch size of 32, a maximum sequence length of 128. For paragraph-level QA datasets (SQuAD, TrivaQA, and NewsQA), we further pre-train BERT for 4 epochs with a learning rate of $5e$-5, a batch size of 16, a maximum sequence length of 384.

Further pre-training s-QUASE. For sentence-level QA datasets (QAMR, Large QA-SRL, and QA-RE), we further pre-train s-QUASE for 64 epochs with a learning rate of $1e$-4, a batch size of 72, a maximum sentence length of 128 and a maximum question length of 24. For paragraph-level QA datasets (SQuAD, TrivaQA, and NewsQA), we further pre-train s-QUASE for 32 epochs with a learning rate of $1e$-4, a batch size of 8, a maximum sentence of 384, and a maximum question length of 64. We need to note that s-QUASE contains more architectures than BERT, so the hyper parameters for BERT fine-tuning are not good for s-QUASE further pre-training

A.2 Error Analysis of QUASE

The F1 scores of s-QUASE$_{QAMR}$ and p-QUASE$_{QAMR}$ on the development set are 76.20 and 90.35. In general, the results of s-QUASE are similar to BiDAF (Seo et al., 2017) but are significantly worse than the p-QUASE on QAMR. We conduct thorough error analysis including: sentence length, answer length, question length, question words and the PoS tag of the answer. We find that s-QUASE is not good at dealing with long sentences compared to p-QUASE. The analy-

sis of model performance with regard to sentence length is shown in Figure 3(a). The average number of QA pairs is much larger when the sentence is longer as shown in Figure 3(b). The distribution of training set and development set is quite different, which makes the situation more complicated. We further compare s-QUASE$_{Large\ QA-SRL}$ and p-QUASE$_{Large\ QA-SRL}$ on Large QA-SRL whose distribution of training and development sets are the same. From the results, s-QUASE is still not as good as p-QUASE on long sentences. We think that the failure of s-QUASE in long sentences is mainly because there are more relations to encode, while p-QUASE only needs to encode information based on specific questions. We believe that there is a trade-off between the quality and the quantity of sentence information that a model can encode in practice, although $h(S)$ also include the information in $h_A(S)$ in a perfect world.

A.3 Ablation Analysis for s-QUASE

s-QUASE consists of three basic components: a sentence encoder for the sentence representation, a question encoder for the question representation, an interaction layer between the sentence component and the question component. We carefully designed five variants of s-QUASE with increasing complexity and performance: (I) Basic model: a fixed BERT and one-layer bidirectional transformer for sentence modeling, a fixed BERT and one-layer bidirectional transformer for question modeling, and a two-layer multi-layer perceptron (MLP) for the interaction layer; (II) a fine-tuned BERT; (III) the same as model II, with a bi-directional attention flow added to the question component; (IV) the same as model III, with the interaction layer changed from a two-layer MLP to a bidirectional transformer; (V) the same as model IV, with the sentence modeling layer and question modeling layer changed from a single-layer bi-directional transformer to a two-layer one, and beam search is used in the inference stage. Table 6 shows the results of our models further pre-trained on the development set of the QAMR dataset.

B Detailed Experimental Setup

In this section, we show the details of experimental setup in Section 3. Because the corresponding settings are too many, we show some common settings here and more details are in our code. We first show the details of simple models in Section B.1,

(a) Error analysis for QUASE.
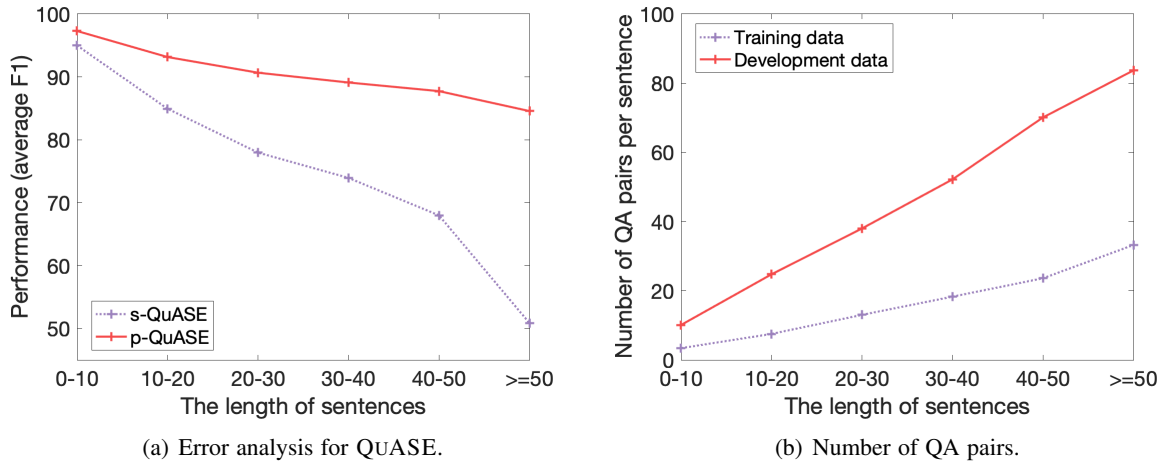

(b) Number of QA pairs.

Figure 3: Error analysis of QUASE on the sentence length. We compare the performance of s-QUASE and p-QUASE on examples with different sentence lengths in the development set. The average number of QA pairs corresponding to the sentence length in the train and development sets is also shown.

Models	Model I	Model II	Model III	Model IV	Model V
Average EM	34.97	41.64	55.68	64.18	**66.77**
Average F1	40.05	45.49	62.98	72.96	**76.20**

Table 6: The results of five variants of s-QUASE$_{QAMR}$ on the development set of QAMR. We use the average exact match (EM) and average F1 as our evaluation metrics.

Embeddings	ELMo				Flair	
Tasks	SRL		Coref		NER	
Split	small	full	small	full	small	full
Baselines	78.32	83.87	60.72	**66.89**	89.86	92.37
s-QUASE$_{QAMR}$	**79.40**	**84.14**	**61.54**	66.58	**90.18**	**92.54**

Table 7: Comparison between s-QUASE$_{QAMR}$ and other STOA embeddings. We use the same experimental settings as Section 3.4 for the three single-sentence tasks, SRL, Coref and NER. We use ELMo embeddings for SRL and Coref, and Flair embeddings for NER as our baselines.

and then show some common experimental settings of QUASE in Section B.2. Finally, we compare s-QUASE with other SOTA embeddings (ELMo and Flair) in Section B.3

B.1 Simple Models

When QUASE is used in the feature-based approach, we need use models for the tasks. For simplicity, we sometimes choose to use some simple models rather than strong baselines in Section 3 in our analysis. Following standard practice, we use a simple BiLSTM model with the input of word embeddings and binary features of predicates for SRL, a simple biaffine model based on BiLSTM for SDP, a simple BiLSTM mode for NER, a simple

CNN baseline with the input of word embeddings and position features for RE, and a simple BiLSTM model for TE.

B.2 Experimental Settings

We use the re-implementation of SRL, NER and Coref from AllenNLP (Gardner et al., 2017) for strong baselines, and we implement the strong baselines of SDP and RE ourselves. As for MRC and TE, we use the re-implementation of BERT with pytorch (Wolf et al., 2019). As for simple models, we implement them by ourselves. As for the hyper parameters for strong baselines of single-sentence tasks, we use the same hyper parameters in the related papers (shown in Section 3). As for the hyper parameters for simple models, we tune them ourselves to find some reasonable hyper parameters. The hyper parameters of MRC and TE for p-QUASE are based on (Wolf et al., 2019).

B.3 Comparison with Other Embeddings

To show whether s-QUASE can also provide extra features than other STOA embeddings[11], such

---

[11]The reported STOA models for SRL and Coref is based on ELMo embeddings and the reported STOA model for NER is based on Flair embeddings.

Sentence	Ann.	Question	Answers
(1) Mr. Agnew was vice president of the U.S. from 1969 until he **resigned** in 1973 .	INF	What did someone resign from?	vice president of the U.S.
(2) This year , Mr. Wathen says the firm will be able to **service** debt and still turn a modest profit .	INF	When will something be serviced?	this year
(3) Mahlunga has said he did nothing wrong and Judge Horn said he "failed to express genuine remorse".	INF	Who doubted his remorse was genuine?	Judge Horn
(4) Volunteers are presently renovating the former post office in the town of Edwards, Mississippi, United States for the doctor to have an office.	IMP	What country are the volunteers renovating in?	United States

Table 8: Some examples of question-answer pairs in QA-SRL and QAMR datasets. The first two examples are from QA-SRL dataset and predicates are bolded. The last two examples are from QAMR dataset. We show two phenomena that are not modeled by traditional symbolic representations of predicate-argument structure (e.g SRL and AMR), inferred relations (INF) and implicit arguments (IMP).

	Span			IOU ≥ 0.5			Token		
Models	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Rules + EM	24.31	22.78	23.52	34.34	32.27	33.27	50.46	28.19	36.17
PerArgument + CoDL + Multitask	32.02	12.30	17.77	46.99	18.06	26.09	70.76	17.80	28.45
Argument Detector + Argument Classifier	**49.19**	**43.09**	**45.94**	**57.84**	**50.82**	**54.10**	**69.37**	**47.60**	**56.45**
Mapping: upper-bound	67.82	48.58	56.61	89.09	65.82	75.70	91.57	70.25	79.50

Table 9: Results of learning an SRL parser from question-answer pairs.

as ELMo and Flair, we compare s-QUASE$_{QAMR}$ embeddings with ELMo embeddings on SRL and Coref, and compare s-QUASE$_{QAMR}$ embeddings with Flair on NER. The results are shown in Table 7. We find that s-QUASE$_{QAMR}$ has a better performance than ELMo and Flair, especially in the low-resource setting, which indicates that s-QUASE can provide extra features than ELMo and Flair.

C On the Strength of Distributed Meaning Representations

In this section, we first show more details of the comparison between QUASE with symbolic meaning representations in Section C.1. After that, we show the details of learning an SRL parser from QA-SRL in Section C.2.

C.1 Comparison with Symbolic Meaning Representations

Probing Analysis. We first show the details of our probing analysis on the Xinhua subset[12] of AMR dataset. Our probing task can be formulated as follows: given two nodes in order, the probing model needs to predict the directed relation from one node to the other. We only consider the cases where there is indeed a relation between them.

---

[12]Only four subsets in AMR dataset contain both training and development sets, but the other three subsets either use informal languages or templatic and report-like structures, which are quite different from the domain of QAMR.

There are 741 sentences and 9008 relations in valid alignments with 70 different types of relations in the training set, and 99 sentences with 1098 relations in valid alignments with 43 different types of relations in the development set. We use the same edge probing model as (Tenney et al., 2019), but we train it by minimizing a softmax loss rather than binary cross-entropy loss. Therefore, our probing results are based on the classification accuracy, not binary F1 score.

Systematic Analysis. We use Large QA-SRL as a testbed to analyze the **representation ability** of p-QUASE$_{QAMR}$. Our p-QUASE$_{QAMR}$ achieves 85.79 F1 score on the development set of Large QA-SRL, while BERT further pre-trained on SQuAD with the same number of QA pairs only achieves an F1 score of 64.63 (it achieves 86.98 F1 on SQuAD). For reference, BERT further pre-trained on Large QA-SRL can achieve 92.19 F1 on Large QA-SRL. All these numbers indicate that p-QUASE$_{QAMR}$ has a strong ability to answer questions related to SRL.

On the other hand, BERT further pre-trained on Large QA-SRL can only achieve 72.17 F1 on the development set of QAMR, while p-QUASE$_{QAMR}$ can achieve 85.79 F1 on Large QA-SRL (it achieves 90.35 F1 on QAMR). These results show that QAMR can cover the questions related to SRL, but Large QA-SRL cannot cover many questions related to AMR. Therefore,

QAMR is a good choice for QUASE to be further pre-trained on.

Some Examples. He et al. (2015) show that QA pairs in QA-SRL often contain inferred relations, especially for why, when and where questions. These inferred relations are typically correct, but outside the scope of PropBank annotations (Kingsbury and Palmer, 2002). This indicates that QA-SRL contains some extra information about predicates. Some examples are shown in Table 8. We further verify that p-QUASE$_{QAMR}$ can correctly answer questions in the examples, which means that QUASE can encode some extra information that SRL cannot.

Michael et al. (2017) show that QAMR can capture a variety of phenomena that are not modeled in traditional representations of predicate-argument structure, including instances of co-reference, implicit and inferred arguments, and implicit relations (for example, between nouns and their modifiers). Some examples of QAMR are shown in Table 8. Similar to SRL, we find that p-QUASE precedes traditional representations, such as AMR, by correctly answering questions in the examples and hence encoding extra information.

C.2 Learning an SRL Parser from QA-SRL

C.2.1 Learng an SRL Parser

We consider learning a SRL parser from QA-SRL. It reduces the problem of learning AMR from QAMR to a simplified case.

Challenges. There are three main challenges to learn an SRL parser from Large QA-SRL.

Partial issues. Only 78% of the arguments have overlapped with answers; 47% of the arguments are exact match; 65% of the arguments have Intersection/Union ≥ 0.5[13].

Irrelevant question-answer pairs. 89% of the answers are "covered" by SRL arguments; 54% of the answers are exact match with arguments; 73% of the answers have Intersection/Union ≥ 0.5. These statistics show that we also get some irrelevant signals: some of the answers are not really arguments (for the corresponding predicate).

Different guidelines. Even if the arguments and the answer overlap, the overlap is only partial.

A reasonable upperbound. We treat the answers that have overlapped with some arguments as our predicted arguments. If two predicted arguments intersect each other, we will use the union of them as new predicted arguments. The results are shown in Table 9. We know from the table that this mapping algorithm achieves a span F1 of 56.61, which is a reasonable upper bound of our SRL system.

Baselines. We consider three models to learn an SRL parser from Large QA-SRL dataset.

Rules + EM. We first use rules to change QA pairs to labels of SRL. We keep the labels with high precision and then use an EM algorithm to do bootstrapping. A simple BiLSTM is used as our model for SRL. The results are shown in Table 9. We think that low token F1 is due to the low partial rate of tokens (37.97%) after initialization.

PerArgument + CoDL + Multitask. We consider a simpler setting here. A small number of gold SRL annotations are provided as seeds. To alleviate the negative impact of low partial rate, we propose to train different BiLSTM models for different arguments (PerArgument) and do global inference to get structured predictions[14]. We first use seeds to train the PerArgument model and then use CoDL (Chang et al., 2007) to introduce constraints, such as SRL constraints, into bootstrapping. At the same time, we train a model to predict the argument type from question-answer pairs. These two tasks (argument type prediction and SRL) are learned together through soft parameter sharing. In this way, we make use of the information from QA pairs for SRL. We use 500 seeds to bootstrap. The span F1 of our method is 17.77 and the span F1 with only seeds is 13.65. More details are in Table 9. The performance of this model has only improved several percents compared to the model trained only on seeds.

Argument Detector + Argument Classifier. Given a small number of gold SRL annotations and a large number of QA pairs, there are two methods to learn an end-to-end SRL[15] system. One is to assign argument types to answers in the context of corresponding questions using rules, and learn an end-to-end SRL model based on the predicted SRL data. This is exactly our first model, Rules + EM. However, the poor precision of argument classification leads to unsatisfactory results. An-

---

[13]These statistics of partial issues and irrelevant question-answer pairs are based on the PTB set of QA-SRL.

[14]Given a predicate in the sentence with three arguments and one of them is annotated, the sentence is partial for a traditional SRL model but not partial for a PerArgument model.

[15]Note that an end-to-end SRL system is with gold predicates. This is different from the generic definition.

other method is to learn from small seeds and bootstrap from large number of QA pairs. Thich is our second model, PerArgument + CoDL + Multitask. However, bootstrapping can not improve argument detection much, leading to mediocre results. We also notice that argument detection is hard with a small number of annotated data, but argument classification is easy with little high-quality annotated data. Fortunately, most answers in Large QA-SRL overlap with arguments. Furthermore, the mapping results of argument detection is about 56.61, good enough compared to two baselines. We propose to learn two components for SRL, one is for argument detection and the other is for argument classifier. We use the span-based model in (FitzGerald et al., 2018) for argument detection. The argument classifier is trained on predicates in the PTB set of QA-SRL. The results are shown in Table 9.

C.2.2 Using SRL/AMR Parsers in Downstream Tasks

There have already been some attempts to use semantics in downstream tasks. We discuss three types of application here. Traditionally, semantic parsers can be used to extract semantic abstractions, and can be applied to question answering (Khashabi et al., 2018). Second, dependency graphs, such as SDP, can be incorporated into neural networks. For example, (Marcheggiani and Titov, 2017) encodes semantic information in Graph Convolution Networks (GCN). In order to use constituent based traditional symbolic meaning representations, one can encode related semantic information by multi-task learning (MTL). (Strubell et al., 2018) mentioned such an example of application.

The main difficulty of retrieving SRL/AMR from QA signals for downstream tasks is to learn a good parser for SRL/AMR from question-answer pairs.