The Effect of Language Representation in Question Decomposition

Gal Suchetzky
galsuchetzky@mail.tau.ac.il

Itay Levy
itaylevy@mail.tau.ac.il

Abstract

A recent paper suggested a new decomposition task of a complex question to a series of simple, natural languages steps that can be executed in sequence for answering the original question. In this paper, we investigate the effect of performing the same task with varying degrees of structured data. We trained several encoder-decoder models using different types of gold data representation. Our experiments show that increasing structurality benefits out-of-distribution generalization significantly.

1 Introduction

1.1 QDMR

In a recent paper by Wolfson et al. (2020) a new decomposition task was introduced, where given a complex question one should express it via simple ("atomic") natural language questions that can be executed in sequence to answer the original question. This formalism is called Question Decomposition Meaning Representation (QDMR). The use of natural language is to make annotation easier for non-experts. Each QDMR sequence can be expressed as a pseudo-logical form of operators and their corresponding arguments.

1.2 Programs

Programs, also called Pseudo Logical forms, are another way of representing the same decomposition as QDMRs in a more structured way. They were also introduced in Wolfson et al. (2020). A program is a series of operators and their respective arguments. See Figure 1 for an example.
The differences between QDMRs and programs are:

1. **Explicitness**: The use of an operator and its respective arguments template in programs (See Table 1 in Wolfson et al. (2020)) explicitly reveals the underlying logic and adds structure, which is absent in the natural-language QDMRs. Furthermore, it makes programs less vulnerable to the ambiguity of natural language.
2. **Anchor tokens**: Programs include special tokens in order to separate the operator from the arguments (@@OP_SEP@@), and arguments from one another (@@ARG_SEP@@). In order to isolate the effect of this difference, we designed and tested another target type which we name "minimized programs". Those are programs without the anchor tokens.
3. **Order**: In programs, unlike QDMRs, the first token (the operator) is also the most salient token. This creates a coarse-to-fine like generation when using a monotonic left-to-right decoder.

Question	what is the current human population of japan ? <eos>
QDMR	return japan @@SEP@@ return human population of @@1@@ <eos>
Minimized program	select japan @@SEP@@ project human population of @@REF@@ @@1@@ <eos>
Program	select @@OP_SEP@@ japan @@SEP@@ project @@OP_SEP@@ human population of @@REF@@ @@ARG_SEP@@ @@1@@ <eos>

Figure 1: Different target types for question decomposition

2 Hypothesis

We hypothesize that neural network based approach will generalize better when decomposing a question to its program rather than to its natural language QDMR. By explicitly predicting operators and arguments instead of predicting natural language expressions, we introduce an inductive bias to our model and make the predictions more structured. Furthermore, we expect performance gaps (the difference in model performance when trained with

different types of data representations) to decrease as models grow in their algorithmic capabilities and in the number of trainable parameters stored in their weights.

3 Experimental Setup

3.1 Approach

We tested our hypothesis by designing a framework that allows conducting controlled experiments where the same model is trained with different target types (QDMRs, programs or minimized programs). We designed several improvements to our basic model which increase its complexity both in algorithmic capabilities and in parameters count.

3.2 Experimentation Framework

For creating the experimentation environment we have adapted and expanded an open-source Pytorch template to support a variety of customizations to models, vocabularies, training, testing procedures and more [1]. Our framework allows using the same model with different configurations that were designed by us to test and compare the different models on both QDMRs and programs.

3.3 Data

We trained models using the training split and evaluated their performance on the validation split of the BREAK dataset from Wolfson et al. (2020).

3.4 Tokenization

We used spaCy tokenizer by Honnibal et al. (2020), which performs word-level tokenization. We added special tokens that are part of QDMRs and programs.

3.5 Model Variations

We have implemented an encoder-decoder recurrent neural model as the base model of our experiments. We implemented several improvements to the basic model, mainly in the decoder module, in order to examine our hypothesis:

3.5.1 Base Model

Our base model is an encoder-decoder architecture with tied weights.

Encoder: We feed the questions, one token at a time, to our encoder which consists of an embedding layer and a single GRU layer.

Decoder: We feed the encoder output and the gold targets (which are QDMRs, minimized programs or programs) to the decoder module, one token at a time. At each decoding step, the decoder embeds a prefix of the gold targets, and uses a GRU cell in order to predict the next target token.

Tied Weights: As shown by Press and Wolf (2017), using the same weights in the encoder embedding layer, the decoder embedding layer and the decoder output layer can benefit the model by introducing regularization. Furthermore, tying weights reduces the number of parameters by a factor of 3. This is helpful when the number of training examples in the dataset is relatively small, which is the case in the BREAK dataset.

3.5.2 Dynamic

Each question in the dataset is paired with a lexicon list. The lexicon of a question consists of all the possible words that might appear in a QDMR of a decomposed question, among other words. The dynamic feature for models restricts the model token predictions only to tokens from the lexicon of the specific question by zeroing the predicted probability of tokens which are not in the lexicon. The fact that all predicted tokens are from the lexicon reduces substantially the rate of token prediction errors. We note that the lexicon was primarily introduced to assist the annotators in the annotation process, and therefore consists of many words that are less relevant for programs (yet still relevant for QDMRs).

Our implementation uses a loss function that masks irrelevant classes (such as token that do not appear in the relevant lexicon) and also irrelevant parts of the target (which are padded using a special pad token).

3.5.3 Attention

We implemented dot-product attention where the queries are the hidden states of the decoder, and the keys and the values are the outputs of the encoder.

3.5.4 Stacked Layers

We added another GRU layer to the decoder.

3.6 Xavier Initialization

All model weights were initialized using Xavier uniform method from Glorot and Bengio (2010).

3.6.1 Dropout

We added dropout layers to the decoder with a dropout rate of 10%.

[1] Open source template for pytorch projects https://github.com/victoresque/pytorch-template

The dropout layer was used after the projection layer of the encoder's output, after the input embedding layer and after the GRU layer.

3.7 Evaluation Methods

3.7.1 SARI Metric

SARI accuracy metric (Xu et al., 2016) is commonly used in tasks such as text simplification. We used the SARI metric in order to evaluate the accuracy of the predictions with respect to the gold decompositions. Given a gold decomposition s, it considers the sets of added, deleted, and kept n-grams when mapping the question x to s. It computes these three sets for both s and the model's prediction $\hat{s}$ using the standard of up to 4-grams, then averages (a) the F1 for added n-grams between s and $\hat{s}$, (b) the F1 for kept n-grams, and (c) the precision for the deleted n-grams. This means that higher SARI values are better.

3.7.2 Unified Metrics

As stated above, we experimented with different representations of question decompositions (QDMRs, programs and minimized programs). In order to compare accuracy scores of predictions in different representations, we decided to compare each of them against a unified gold decomposition using SARI metric. We chose the form of minimized programs as the unified form, due to the fact that it stores the least amount of information and variability, relative to other decomposition forms. This means converting QDMRs to programs (using existing rule-based method from Wolfson et al. (2020)) and then converting programs to minimized programs. Any other choice of a unified form would require a non-trivial expansion process that includes necessary information from the original question as well. We discuss the possible problems of this approach in section 6.

We chose the SARI metric because it handles prediction errors better then other metrics such as exact match. Intuitively, the more matching n-grams we have in the prediction and in the gold decomposition, the more likely it is that the prediction represents the same idea as the gold. Moreover, it handles conversion errors better then the other metrics because it is less vulnerable to local differences.

3.8 Reproducibility

The configurations and hyper-parameters we used are documented in our repository. Each experiment was conducted with a different configuration json file available there.

3.8.1 Computational Cost

Conducting our experiments took 220 hours on NVIDIA GTX 1070 Ti type GPU.

4 Experiments

4.1 Effect of Model Capabilities on Performance Gap

Our first experiment was designed to examine the effect of increasing model capabilities on performance in each setting. As evident from figure 2, the performance gap between program and QDMR prediction roughly decreases as the model has more algorithmic features.

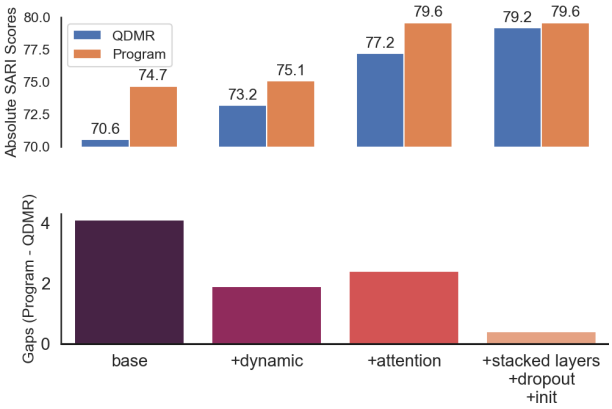

Figure 2: SARI absolute and relative (gaps) scores of models with gradually increased capabilities. The upper figure shows the absolute SARI scores of QDMR prediction and program prediction. Consistently, program prediction scores higher. The lower figure shows the difference in scores between the two.

4.2 Out of Distribution Generalization

We conducted several experiments in which we trained and evaluated the models on different data distributions. We chose distributions in a fair manner by verifying they have roughly the same probability mass in the BREAK dataset. All experiments were conducted with varying decoder hidden size over the best model of section 4.1. All experiments were conducted when training with QDMRs, programs and minimized program. Figure 3 shows the vanilla baseline for those experiments where the models were trained on the whole training split and evaluated on the whole evaluation split. This experiment shows there is no clear a-priory advantage to training the models with programs vs. QDMRs.

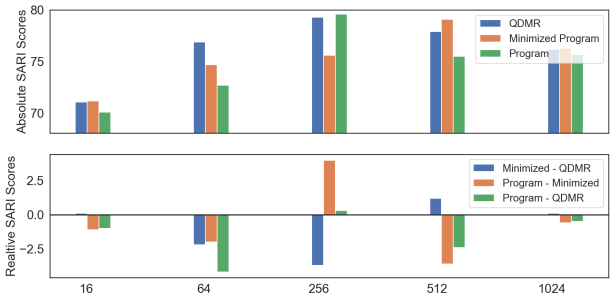

Figure 3: SARI absolute and relative scores of models with gradually increased decoder hidden size in the **vanilla baseline** experiment. The upper figure shows the absolute SARI scores. The lower figure shows the difference in scores between pairs of decomposition representations.

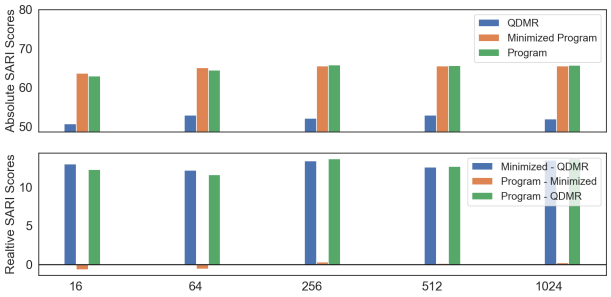

Figure 4: SARI absolute and relative scores of models with gradually increased decoder hidden size in the **domain split** setting. The upper figure shows the absolute SARI scores. The lower figure shows the difference in scores between pairs of decomposition representations.

4.2.1 Domain Adaptation Ability

BREAK dataset consists of a collection of questions from ten QA datasets of three modalities - texts, images and databases. 51% of the questions in BREAK dataset have a text modality such as "What was Gandhi's occupation before becoming a freedom figther?". Those questions originate from the COMQA, CWQ, DROP and HOTPOTQA-HARD datasets. the rest 49% of the questions have images modality (33%) or database modality (16%). Images modality questions, such as "Are there only two dogs pulling one of the sleds?", originate from the CLEVR-HUMANS or NLVR2 datasets. Databases modality questions, such as "How many transactions correspond to each invoice number?", originate from ACADEMIC, ATIS, GEOQUERY and SPIDER datasets.

In this experiment we trained each model only on questions of text modality in the train split by filtering the origin dataset field. We evaluated the model only on questions of images and databases modalities from the evaluation split. We repeated this process using different decoder hidden size. This way, we were able to test the ability of each model to generalize to questions outside the modality or domain it was trained on. Figure 4 shows a significant and consistent advantage to programs or minimized programs based models in this setting, regardless of the hidden size. the difference in performance between program based models and QDMR based models is always above 11 SARI points.

4.2.2 Evaluating on Longer Questions

56% of the QDMR annotations in BREAK datasets have 1-4 steps and we refer to them as short annotations. The remaining 44% of the QDMR annotations have at least 5 steps and we refer to them as long annotations. We trained each model on questions that have only short annotations and evaluated it on questions that have only long annotations.

Figure 5 shows a significant and consistent advantage to program or minimized programs based models in this setting, regardless of the hidden size. The difference in performance between program based models and QDMR based models is always above 18 SARI points.

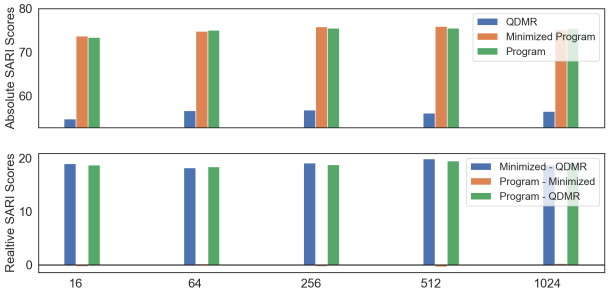

Figure 5: SARI absolute and relative scores of models with gradually increased decoder hidden size in the **length split** setting. The upper figure shows the absolute SARI scores. The lower figure shows the difference in scores between pairs of decomposition representations.

5 Discussion

Our experiments show a clear advantage of training a model with programs over training it with QDMRs, even when the annotation process was conducted with QDMRs. It significantly improves out-of-distribution generalization in all splits studied, regardless of the decoder hidden size. Comparing the results of the domain adaptation and length adaptation experiment with the vanilla baseline ex-

periment highlights the adaptability of program predicting models to out-of-distribution examples in contrast to QDMRs predicting models.
The indifference of this phenomena to the change in model size does not agree with the second part of our hypothesis. However, our hypothesis is supported in the case of in-distribution generalization, which according to our results benefits from training with programs especially in low-resource scenarios where complex models cannot be applied.
The difference in performance between programs and minimized programs is negligible across multiple experiments. This means that the addition of anchor tokens in programs plays a small role in making them an effective form for question decomposition.

6 Limitations

Studying the difference between QDMRs and programs might be complex and not trivial. Here we list some of the problems and biases we have encountered while designing the experimental setup and evaluation methodologies:

1. **QDMR to program conversion failures** might introduce bias. As the conversion procedure is a rule based process, it might fail for bad predictions that do not fit any conversion rule. QDMR conversion failures might heavily impact the calculated accuracy of the model (while the model itself might predicting QDMRs reasonably well). Therefore, we are biased towards the program prediction task. We dealt with it separately in each experiment:

    - In the varying complexity experiment (section 4.1) we mainly studied how the gap in performance of programs vs. QDMRs changes as the model becomes more capable.
    - In the out-of-distribution experiments (section 4.2) we compared our result to a vanilla baseline that shows no clear preference for programs over QDMRs.

    Another approach to solve these issues might be using a neural conversion model that deals with the differences in a finer way.

2. **Difficulty of recurrent decoder to process long sequences**. A unidirectional decoder is limited in its ability to ”remember” tokens from the beginning of a long sequence, therefore the model that is predicting the longer sequence will perform worse. In order to mitigate this problem we checked and verified that QDMRs and programs have roughly the same number of tokens when using spaCy tokenizer (less than 1% difference).

3. **Only recurrent based networks has been studied**. Transformer based models might perform even better and further prove the hypothesis.

4. **Hyper-parameter tuning was not conducted**. Due to lack of time, as the experimentation process was already very long (see section 3.8.1), we did not perform hyper-parameter tuning.

5. **Using other evaluation metrics**. We note that additional graph and normalization based metrics (GED and normalized EM) could potentially be used to increase the validity of our results, but major alterations are required for them to successfully measure the accuracy of programs.

7 Related Work

Multiple studies on semantic parsing (Kate et al., 2005, Liang et al., 2011, Guo et al., 2019) showed that using an appropriate meaning representation can substantially improve the performance of a semantic parser.

A recent paper by Guo et al. (2020) showed the problematic effect of program aliases (semantically equivalent but syntactically different logical forms) on meaning representations. This it also apparent in our study, where natural language QDMRs have more degrees of syntactic difference compared to programs, which makes it difficult to predict them precisely.

As stated above, predicting a program step begins with predicting the operator (the most salient token in each step) and continues to filling the missing arguments. This process resembles a coarse-to-fine decoding mechanism. Dong and Lapata (2018) studied the benefits of such approach for semantic parsing.

8 Conclusion

In this work we designed several experiments for thoroughly examining the effect of training a seq-

2-seq neural network with different forms of questions decomposition. Those experiments show the benefits of using programs, a more structured form, relative to QDMRs both in the in-distribution setting and the out-of-distribution setting.

In-Distribution Generalization: Training models to predict programs is preferable when limited compute resources are available and complex models cannot be applied.

Out-Of-Distribution Generalization: The results of our experiments indicate that program predicting models adapt significantly better to out-of-distribution examples then QDMR predicting models.

Therefore, it is advised to pre-process the data fed to the model into more structured forms in order to achieve better generalization in question decomposition tasks.

9 Future Work

In this work, we showed that programs are preferable to QDMRs in some settings. However, many more experiments and testing can be done to further test the hypothesis proposed:

- In Wolfson et al. (2020), the authors show that open-domain question answering neural that exploit QDMRs perform better than when they use only the original complex questions. A natural question arising from that result and our study is whether replacing QDMRs with programs would be even more beneficial for this task.

- Another line of work could thoroughly examine the effectiveness of each operator independently. Also, we suggest developing a method for learning the operators and their arguments on-the-go instead of using a fixed vocabulary of predefined operators.

- As stated above, the operators in programs resemble a coarse-to-fine approach where the most salient part of each step is decoded first. One could potentially expand this approach beyond individual steps to the whole decomposition level by using non monotonic insertion-based decoding and thereby decoding the most salient operators across all steps first.

Acknowledgments

We want to thank Matan Hasson (Tel Aviv University) for kindly answering our questions about evaluation metrics. We also thank Chris Coleman (Northwestern University) for explaining how to reproduce his results from the BREAK leaderboard.

References

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Jiaqi Guo, Qian Liu, Jian-Guang Lou, Zhenwen Li, Xueqing Liu, Tao Xie, and Ting Liu. 2020. Benchmarking meaning representations in neural semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1520–1540, Online. Association for Computational Linguistics.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Rohit J. Kate, Y. W. Wong, and R. Mooney. 2005. Learning to transform natural to formal languages. In *AAAI*.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon, USA. Association for Computational Linguistics.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *EACL*.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. BREAK It Down: A Question Understanding Benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.