Advancing Neural Question Generation for Formal Pragmatics: Learning when to generate and when to copy

Haemanth Santhi Ponnusamy, Madeeswaran Kannan,
Kordula De Kuthy, and Detmar Meurers
SFB 833, Project A4
University of Tübingen

Abstract

Question generation is an interesting challenge for current neural network architectures given that it combines aspects of language meaning and forms in complex ways. The systems have to generate question phrases appropriately linking to the meaning of the envisaged answer phrases, and they have to learn to generate well-formed questions using the source.

Complementing the substantial strand of research on question generation in application contexts, some recent work also highlighted the role that questions and question generation can play conceptually in formal pragmatics for linking the information structure of sentences to the discourse structure of texts in so-called Question-under-Discussion (QuD) approaches (De Kuthy et al., 2020).

In this paper, we show that the sequence to sequence architecture employed in the previous work fails to capture a key property of the task: the required question-answer congruence ensures that the lexical material needed for the question is explicitly given by the answer generated from. Extending the architecture with a pointer component helps overcome this shortcoming. Second, we enrich the model with part-of-speech and semantic role information to improve question phrase generation. The resulting approach quantitatively advances the state of the art in terms of BLEU scores and question well-formedness, and we qualitatively discuss key linguistic characteristics of the generated question.

1 Introduction

Question generation, the task of creating natural questions for a given sentence or paragraph, is a challenging task with potentially many practical applications – from question answering, via dialogue systems, to reading comprehension tasks. Following the first, rule-based QG systems, the recent state-of-the-art approaches are generally based on neural networks. The task of QG is typically formulated as a sequence-to-sequence (seq2seq) learning problem in which a sentence is mapped to a corresponding question (cf., e.g., Pan et al., 2019).

In formal pragmatics, questions also play an increasingly prominent role in so-called Questions-under-Discussion (QuD, Roberts, 2012; Velleman and Beaver, 2016) approaches. Here, questions are used to make explicit the interface between the information structure of a sentence and the particular discourse structure that the sentence can function in. Under this QuD perspective, for every sentence in a text, a question needs to be formulated – and indeed explicit guidelines have been defined to support reliable manual QuD annotation (Riester et al., 2018). In a recent paper, De Kuthy et al. (2020) argue that such question generation should be automated to support the analysis of large corpora, and they propose a seq2seq neural network approach to generate all potential questions for a given sentence. Their approach learned to (often) predict the correct question word for a given answer phrase and generated questions that correctly reflect the word order properties of questions in German.

While this result confirms that neural networks can be successfully trained to generate meaningful, well-formed questions to pursue a formal pragmatic vision, there are clear challenges for such a seq2seq architecture that is supposed to generate questions for any type of large data set. One problem are rare or unknown words that have to be predicted. In most neural generation architectures, words are the basic input and output tokens. Pretrained word embeddings are used to initialize the token embedding matrix and generally a fixed vocabulary is used for both input and output sequences. With a restricted vocabulary, given the Zipfian distribution of words in language use, in any authentic corpus material serving as input there are likely to be rare

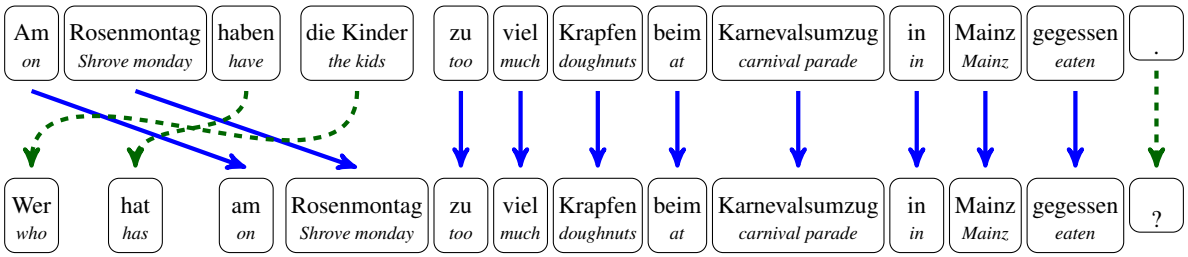

Figure 1: An example showing identical words in source sentence and question (with solid blue links) and the question word and subject-verb agreement requiring changes in the question formulation (dashed green relation)

or unknown words that are not part of the fixed vocabulary and therefore cannot be predicted in the output layer, the generated question. This indeed is a major issue mentioned for the question generation approach of De Kuthy et al. (2020). To overcome this problem, they implemented an ad-hoc post-processing step: All generated questions are checked for markers indicating the places where an out-of-vocabulary token appears. A heuristic then tries to identify that missing word in the source sentence and insert it in the right place of the output.

When we conceptually consider the task of question generation from source sentences, this is a problem that should not arise – after all, the source sentence is explicitly provided and the words in the question to be generated can be selected from that source material, to which the question words, which can be drawn from a fixed set of language expressions for a given language, need to be added. So the task of generating a question based on a given sentence conceptually consists of two subtasks: (i) Identifying the material that is identical between source sentence and question and can simply be copied over, and (ii) predicting the new material appearing in the question, in particular the correct question words. This is illustrated by the sentence-question pair in Figure 1. In that example, the specialized carnival terminology, *Karnevalsumzug* and *Rosenmontag*, are typical rare words, and the use of the city name *Mainz* illustrates the occurrence of named entities.

A related problem has been discussed in recent work for question generation (Zhao et al., 2018) and for text summarization (See et al., 2017). (Zhao et al., 2018) propose to extend a seq2seq attention model with a pointer mechanism in order to improve their task of paragraph-level QG. They show that their model with the copy mechanism can learn to generate some words in a question and to copy others from the input text. But they do not provide any details about which parts of the question are copied and which are generated and in which way the copy mechanism improves the generated questions. (See et al., 2017) observe that baseline seq2seq models for summarization often replace an uncommon (but in-vocabulary) word with a more common alternative and fail to reproduce out-of-vocabulary words. They show that the copy mechanism of their pointer-generator model overcomes both these problems, but again they do not provide any details based on which information the model chooses to copy or generate the different parts of the sentences in the summarization.

In this paper, we adopt a pointer-based architecture for the generation of questions in pursuit of the formal pragmatic vision of generating QuDs for every sentence in a text. Such an architecture turns out to be more successful than the seq2seq based model, replacing the ad-hoc heuristic post-processing step used in previous work into a design feature of the neural network architecture. In architecturally separating the copying from the generation component, it also supports the integration of further linguistic information needed to successfully determine which parts of the sentence can be copied over to the question and which parts have to generated, as for examples the question phrase.

For the mentioned example, Figure 2 identifies the minimal case, i.e. the rare or unknown words that should be copied using the pointer component, whereas other words can or need to be generated to fit the output context, such as the question word *wer (who)* and the subject-verb agreement that needs to be adjusted from plural *haben (have)* to singular *hat (has)*. We will show in a detailed analysis of the attention scores that our model learns to generate material in the question only in four cases: question word, question mark, lower-cased first word and verb form. All other material from the source sentence are copied over to the question.

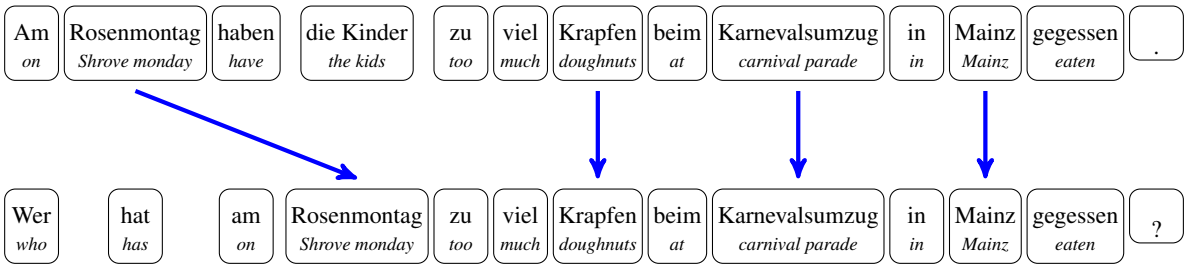

Figure 2: Example illustrating minimal identification of rare words and named entities in support of QG

The paper is structured as follows: After providing some background on the different architectures used for neural QG, in section 3 we spell out the specifics of the question-answer data we use to train the different QG models. In section 4, we present our neural QG models. As a baseline, we train a seq2seq model, similar to the one presented in (De Kuthy et al., 2020) and compare this to two versions of a pointer-based neural model, a baseline model and a model extended by two word-level features, POS tags and semantic role labels. The methods are all evaluated in quantitative terms using BLEU scores. For a qualitative analysis, we verify for how many of a random set of 500 sentences the different models produce how many well-formed questions and provide some examples illustrating the pros and cons of the different models.

2 Related Work

Generating questions is a challenging task regardless of the language. Prior to the advent of deep neural networks, question generation was largely restricted to transformation-based methods that leverage linguistic characteristics and syntactic structures (Liu et al., 2010; Curto et al., 2012; Heilman, 2011). These methods are inherently limited in that they are designed to generate questions based on pre-programmed rules that manipulate parse trees and scale very poorly with linguistic complexity. Deep neural methods, on the other hand, learn - from large sets of data - latent representations of syntactic and semantic language characteristics, amongst others.

Framing question generation as a sequence learning task enables one to exploit sequence-to-sequence architectures (Sutskever et al., 2014) that have seen significant success in neural machine translation. Sequence-to-sequence (*seq2seq*) architectures consist of two networks: 1) an encoder network that learns a representation of the source sequence, and 2) a decoder network that generates target words one at a time. This architecture was improved upon by incorporating local and global attention mechanisms (Bahdanau et al., 2014; Luong et al., 2015) that modulate the contribution of each token (in the source sequence) to the tokens in the target sequence.

A recent survey of neural question generation research (Pan et al., 2019) shows that the above-mentioned architectures form the basis of many NQG models. Du et al. (2017) condition a generative model on target answers by encoding the position of the answer in the context as an input feature. Sun et al. (2018) split the QG task into first determining the question type and then generating the question using a template-based approach with two *seq2seq* models. Kumar et al. (2018) leverage linguistic features such as POS and NER tags and deep reinforcement learning techniques such as policy gradient methods to add additional task-specific rewards to the training objective. Rare words present a challenge to generative NLP models, and NQG models are no exception. Gulcehre et al. (2016) propose a neural model for machine translation that uses a MLP in tandem with dual softmax layers to determine when to predict a word from a fixed vocabulary and when to point to one in the source sentence. Gu et al. (2016) and See et al. (2017) showed the efficacy of pointer-generator networks at the task of abstractive text summarization. Zhao et al. (2018) adapt the same network by augmenting the encoder with gated self-attention and the decoder with a maxout pointer mechanism to deal with larger contexts. While all these implementations of pointer-based mixture models exemplify their ability to solve the unknown word problem and the advantage of copying words from the context, it is still unclear how exactly the model adapts to the task at hand, i.e., how the competing generator and pointer networks contribute to the final score and under what circumstances one is preferred over the other.

We employ the task proposed by De Kuthy et al. (2020), in which question generation is defined in the context of the formal pragmatic QuD approach where a question is generated from every sentence in a given text. We replace their sequence-to-sequence model and post-process copy step with a unified pointer-generator network that significantly outperforms the former. We also provide detailed insight into the generation and copying characteristics of the latter.

3 Data

Since question generation has primarily been approached as a sub-task of question answering, a large majority of the relevant corpora are generally tailored as QA datasets first and foremost. While datasets such as SQuAD (Rajpurkar et al., 2016), Coqa (Reddy et al., 2019), Quac (Choi et al., 2018) can nevertheless be used to train and evaluate pure question generation models, they unfortunately come up short in the context of our task.

The most obvious downside to datasets such as the above is that nearly all of them are exclusively available in English. The few that are multilingual such as XQUAD (Artetxe et al., 2019) and MLQA (Lewis et al., 2019) are too limited in size to be used for training neural models since they are originally intended to be used as evaluation datasets for question generation/question answering models. This limitation, however, is not insurmountable; one could leverage machine translation[1] to automatically translate one of the above corpora to the target language to create a potentially usable dataset. An alternative, more active approach could involve developing a neural model that is able to jointly translate, align and generate questions (Carrino et al., 2019). Unfortunately, both approaches have a significant disadvantage in that their outputs can be expected to be of significantly lower quality then human-generated output. This can in turn increase the potential of affecting the model's performance in the actual downstream task of question generation due to the increased error propagation at the translation stage.

The second downside to using corpora such as SQUAD is that they are designed to provide paragraph-level contexts for questions. Each question can potentially have multiple ground-truth answers that can be spans of any sentence in the context. This fundamentally changes, i.e., decreases the Q-A-Congruence of the question-answer pair, making them unsuitable for the generation of assertion-level questions, as is our goal here, following the approach proposed in (De Kuthy et al., 2020) for the generation of sentence-level QUDs.

Given the above-mentioned limitations of using pre-existing QA corpora, we obtained the German QA answer corpus descrined in (De Kuthy et al., 2020). This corpus of 5.24 million sentence-question-answer triples is based on sentences from the German newspaper *Die Tageszeitung (taz)* [2] and questions were generated using the only available comprehensive transformation-based question generation system (Kolditz, 2015) for German.

Due to inherent limitations of transformation-based approaches to question generation, such systems are not always capable of producing a question for a given sentence. Furthermore, the system in question only contains a limited domain of transformation rules that mainly selects NPs and PPs as answer phrases and transforms sentences into wh-questions asking about subject and object NPs and several types of PP adjuncts and adverbial phrases. The example in (1) illustrates some types of questions and answer phrases that are produced by the transformation rules and that are part of the QA corpus created by (De Kuthy et al., 2020).

(1) a. Die Kinder essen am Sonntag Kuchen im Garten.
The children eat cake in the garden on Sunday.
b. Wer isst am Sonntag Kuchen im Garten. - Die Kinder
Who eats cake in the garden on Sunday - the children
c. Was essen die Kinder am Sonntag im Garten? - Kuchen
What do the children eat in the garden on Sunday? - cake
d. Wann essen die Kinder Kuchen im Garten? - am Sonntag
When do the children eat cake in the garden? - on Sunday
e. Wo essen die Kinder am Sonntag Kuchen? - im Garten
Where do the children eat cake on Sunday? - in the garden

4 Neural Question Generation Architectures

The task of question generation is formulated as a sequence learning problem where given a source sentence or context as the input sequence $x_1, ..., x_n$ and a target answer phrase a, the model learns the conditional probability $p(y|x, a)$ of generating the

[1] https://cloud.google.com/translate

[2] https://taz.de/

target question $y_1, ..., y_m$:

$$log\ p(y \mid x, a) = \sum_{j=1}^{m} log\ p(y_j \mid y_{<j}, x, a)$$

spaCy's `de_core_news_sm` model was used for parsing and tagging the input sentences for both models. Answer phrase spans were encoded in IOB format. *fastText* embeddings (Bojanowski et al., 2017) were used as pre-trained token embeddings. Input and target vocabulary sizes were fixed to 100K most frequent words in the corpus.

4.1 Sequence-to-sequence Model

The baseline *seq2seq* model is identical to the one used by De Kuthy et al. (2020). The input sequences to the model are the source sentence's word tokens, their part-of-speech tags, and the answer phrase span. Since this architecture does not implement an explicit mechanism to handle out-of-vocabulary words, an ad-hoc post-processing pass is performed on the model's predictions to automatically resolve OOV tokens by locally aligning the parses of the source sentence and the predicted question.

4.2 Pointer Model

Our pointer model is an extension of the work done by Zhao et al. (2018), who implement a Maxout pointer mechanism with gated self-attention.[3]. We experimented with two variants of input sequences. In the first variant, the input sequences were restricted to surface form tokens of the source sentence and the span of the answer phrase. Then we added the parts of speech (POS) and semantic role labels (SRL) in the next variant. Canonical representations of the encoder variants are shown below:

$$u_t = RNN(u_{t-1}, [e_t, a_t]) \quad (1)$$

$$u_t = RNN(u_{t-1}, [e_t, a_t, p_t, s_t]) \quad (2)$$

e_t is the embedded word tokens of the source sentence, a_t answer tagging embedding, p_t represents the POS embedding and s_t indicates the embedded semantic role labels. Now the encoder hidden state u_t is computed through the function of previous encoder hidden state u_{t-1} and the concatenated feature embeddings $[e_t, m_t]$ or $[e_t, a_t, p_t, s_t]$. Further, the hidden states $\{\widehat{u}_t\}_{t=1}^{M}$ are refined using the self-attention. The raw attention scores (Luong et al., 2015) computed between the encoder hidden state U and the decoder hidden state d_{t-1} are used to compute the copy scores. The general approach of the copy mechanisms is to treat each word in the source sentence to be a unique target to point to and to compute the scores separately. In the end, the scores of the words that occur repeatedly in the source sentence are added to get a final copy score. This leads to an overshoot of the copy scores for the words that are repeated in the source, resulting in repeated predictions of the same in the target sequence. To overcome this issue, only the maximum copy score of each word is used (Goodfellow et al., 2013; Zhao et al., 2018). An expression of the scoring mechanism is shown below:

$$sc^{copy}(y_t) = \begin{cases} \max_k r_{t,k}, & y_t \in \chi; x_k = y_t \\ -inf, & otherwise \end{cases}$$

x_k is the k^{th} word in the source sequence and y_t is the t^{th} word in the output sequence. χ is the vocabulary of all words in the input sequence, and $r_{t,k}$ is the raw attention score between x_k and r_t.

The scores from the copy mechanism and the decoder are softmaxed and combined to get the final probability distribution over the extended vocabulary containing the OOV tokens. Since the raw copy and generation scores are added together as a single vector, the copy module and the generation module essentially compete with each other for the final prediction at each timestep.

Hyperparameter	Value
Batch size	64
Epochs	10
Encoder RNN Unit	Bi-LSTM
Decoder RNN Unit	LSTM
Encoder/Decoder Hidden Size	300
Encoder/Decoder Dropout	0.3
Word Embedding Dim	300
Answer Span Embedding Dim	3
POS Embedding Dim	25
SRL Embedding Dim	25
Min Decode Step	8
Max Decode Step	100

Table 1: Pointer Model Hyperparameters

5 Evaluation

The *seq2seq* model and the pointer network were trained on the same 400K training samples. Validation and test sets were set to 15K samples each. For quantitative evaluation, questions predicted by the

[3]Unofficial implementation: https://github.com/seanie12/neural-question-generation

models were compared to the ground-truth questions from our QA corpus and their corresponding BLEU (Papineni et al., 2002) scores were calculated (Table 2).

Even though the *seq2seq* models lack a copy mechanism in their architecture, they adequately learn to mimic the behaviour by positively biasing the generative probabilities of (in-vocabulary) words that appear in the source sequence. The post-processing copy operation, though error-prone, extends this to out-of-vocabulary words, improving performance even further. In contrast, the pointer models unequivocally show that implementing copying directly in the neural architecture improves performance even in the absence of additional linguistic features such as part-of-speech tags and semantic role labels.

5.1 Model Comparison

The high BLEU scores for all of our models indicate that the models are all capable to learn the task of generating questions. To investigate where the differences and particular strengths of the different models are, we provide a more in-depth qualitative analysis of the three models. We performed a manual evaluation of a random set of questions produced by all our models for the same set of sentence - answer phrase pairs. The sample set was obtained by randomly sampling 500 sentences from the original TAZ corpus. For the comparison of our three question generation models, the 500 sentences plus the answer phrases from the rule-based output described in section 3 were used. Based on this set of 500 sentences plus answer phrases, the three neural QG models generated 500 questions each, i.e., one question per sentence - answer phrase pair. Next, the quality of the generated questions was manually evaluated by two human annotators with good annotation agreement ($\kappa = 0.74$), i.e., whether a question is well-formed and whether there is question-answer congruence between the question and the source sentence.

For the 500 questions generated by each model, the baseline seq2seq model shows the worst performance with only 31% well-formed questions out of 500. Adding the post-processing step of replacing OOV words by a word from the source sentence increased the number of well-formed questions to 52%. The two pointer architectures produced well-formed question with improved accuracy: The baseline pointer model produced 55% well-formed questions, while the pointer model with POS and SRL features produced 61% well-formed questions, the best performance for this sample set. The table in 3 sums up the results of this evaluation.

Table 4 shows a systematic analysis of the most frequent errors in the 500 sample questions made by the three models. One can, for example see, that while the questions from seq2seq model still contained unknown words in 47 cases (even after the post-processing), the questions of both pointer models did not have this problem anymore.

5.2 Copying vs Generation

The pointer model with attention and a copy mechanism successfully learned to point to the OOV tokens from the input string and copy them over to the predicted question. The generated questions thus do not contain any OOV tokens anymore. What is not obvious right away is whether the pointer architecture also learned to point and copy over other parts of the sentence and to generate only where really necessary in order to produce a new form. An investigation of the raw attention scores used to compute the copy scores that determine whether a token can be copied over between input and output or needs to be generated showed that indeed the model learned to simply copy over many parts of the source sentence into the question. Figure 3 shows a typical sentence-question pair from our 500 sample, containing 4 instances of generated tokens: *Wer* question word, *hält* Infinite verb to match the subject, *deshalb* lower case transformation to the first word and *?* question mark.

Figure 4 shows the softmaxed scores of the attention between the previous decoder hidden state at every timestep to all the encoder hidden states. Each column here shows the distribution of weights corresponding to hidden representations of each word in the input sequence towards the computation of the context vector. The output token at that time step is produced as the result of the function of this context vector and the previous decoder hidden state. The tokens with the highest attention scores in each column indicate the primary focus of the model before generating the respective output. Since the words *Wer, hält, deshalb and ?* are generated in the output and not copied, we can infer that the hidden states corresponding to the highest scores in each column have a direct influence in generating these words. The higher attention on the word *Professor* in the input sequence to gener-

Model	Training Size	Features	BLEU-1/2/3/4	Cumulative
seq2seq	500k	Word, Ans, POS	84.9/75.0/67.1/60.3	71.25
seq2seq + Copy	500k	Word, Ans, POS	93.8/86.5/81.0/76.5	84.24
Pointer	500k	Word, Ans	97.0/91.0/86.7/83.4	89.40
Pointer	500k	Word, Ans, POS, SRL	98.0/92.9/89.1/86.3	**91.45**

Table 2: Evaluation results

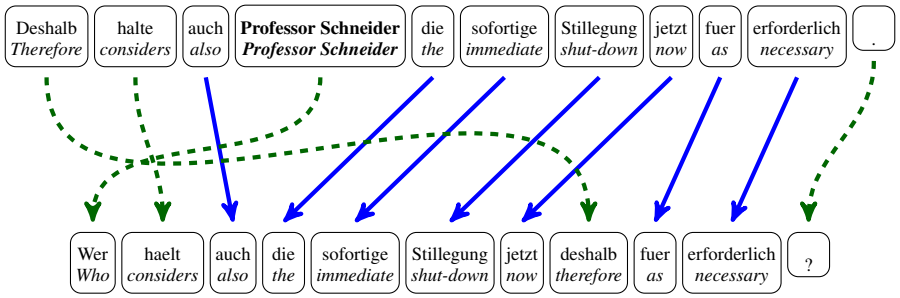

Figure 3: A question generation example, highlighting copy and generate decisions

Model	Well-formed Questions
Baseline seq2seq	31%
seq2seq + Copy	52%
Baseline Pointer	55%
Pointer + Ling. Features	**61%**

Table 3: Results for random sample of 500 sentences

Error Type	Seq2Seq	Ptr1	Ptr2
Question word	88	105	88
Unknown Word	47	0	0
Word Order	40	29	24
Different Word	18	31	15
Missing Word	6	7	6
Verb Form	6	7	5

Table 4: Distribution of error types in the 500 samples

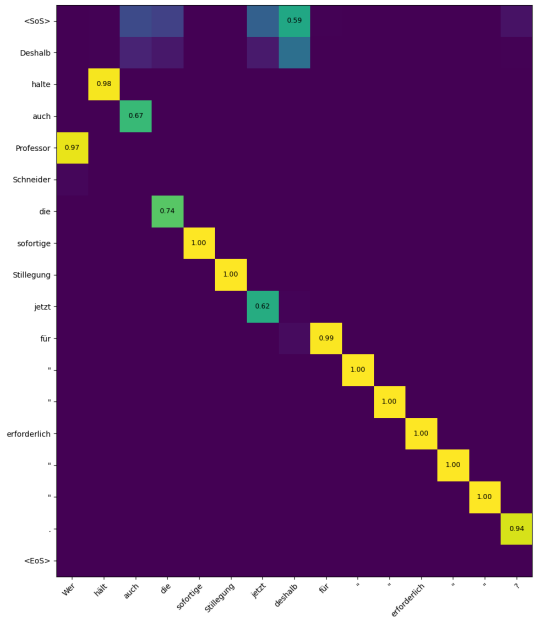

Figure 4: Softmaxed attention weight used for computing the context vector as input to each decoding step

ate the appropriate question word *Wer* shows that the model has learned the relationship between the nature of answer phrase *Professor Scheider* and the type of the question phrase.

To illustrate how the model uses information from the attention scores in the decoding step and also to compute copy scores, Figure 5 shows the raw attention scores between the previous decoding hidden state at every timestep to each of the encoding hidden states corresponding to the input tokens. The maximum scores in each column directly correspond to the score used by the copy module to compete with the generated scores. The streaks of high scores as diagonals shows that a chunk of the source sentence is copied with high support from the attention. This behaviour of replicating most of the information from the source sentence instead of generating new tokens shows that the model has adapted to the nature of the task including the right decision between copying or generating based on linguistic features.

We can now also precisely determine how often the pointer model is generating and copying and what type of tokens are being generated. As shown above, the decision for predicting each word in the output sequence is influenced by their intermediate scores. We can thus categorize the decisions into four categories: *Copy* - Only the copy module has suggested the final prediction with high confidence, *Generate* - Only the generate module has

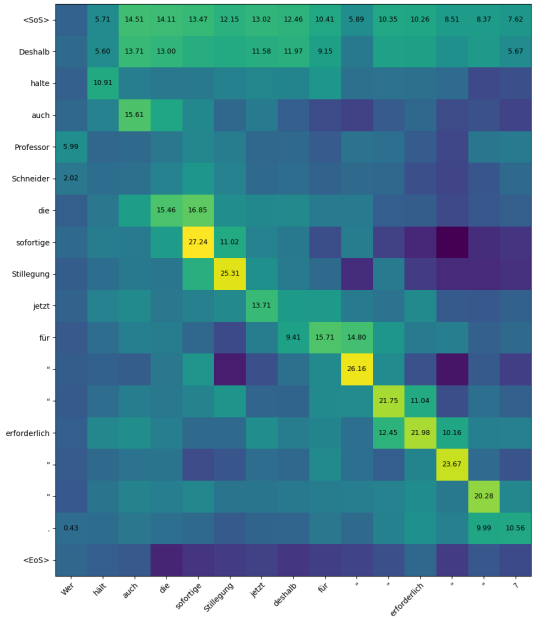

Figure 5: Raw decoder attention scores used directly as the copy scores

suggested the final prediction with high confidence, *Both* - Both the modules has suggested with high confidence and *Neither* - Neither of the modules suggested the final prediction with high confidence but jointly achieved the final prediction.

Category	Avg. % of a question
Copy	79.32%
Generate	17.57%
Neither	2.12%
Both	0.48%

Table 5: Direct influence of the modules on the final prediction of each question

Table 5 shows that around 79% parts of the questions are being copied and only around $17 - 18\%$ being generated. 2% parts of the question are being jointly predicted by both the generation and the copy modules and just less than 1% are mutually agreed by both the modules. We also determined that the model only generates tokens in four cases: Question word, question mark, lower-cased first word and verb form.

5.3 Greedy vs Beam search

We here briefly discuss the effect of different sequence search strategies like beam search vs the greedy approach to achieve a balance between the quality of the generated output and the computational cost. It has been observed that the beam search strategy might not be advantageous in all the cases of sequence generation. (Cohen and Beck, 2019) highlighted the effect of degrading performance of the sequence generation models with the increase in beam width. In our model, we face a similar scenario, where the increase in beam width during the decoding stage harms the model's performance both quantitatively and qualitatively.

Beam width	BLEU (1/2/3/4)	(Cummulative)
1 (Greedy)	98.0/92.9/89.1/86.3	91.45
3	96.7/91.2/87.4/84.5	89.83
5	96.0/90.4/86.4/83.5	88.94
7	95.7/90.0/85.9/83.1	88.54
15	95.3/89.5/85.4/82.5	88.05

Table 6: Degrading effect of beam width on the pointer model's performance

In Table 6, the model version with the greedy search (beam width=1) approach performs much better than the other versions with the increased beam width. This behaviour is due to the nature of our task, which requires predominantly the exact words to be copied from the source sentence into the output sequence. As we have shown above, the model prefers to copy around 79% of the question with very high confidence. So choosing an alternative for the exact words that are supposed to be copied and choosing words that maximize the overall probability in subsequent steps lead to a mispredicted sequence. This error mainly happens when the copy module suggests the words with relatively lesser confidence.

6 Conclusion and Outlook

Given the task of question generation in a formal pragmatics context, we successfully trained and tested two different neural network architectures on a dataset of natural question-answer pairs from a German newspaper corpus. We showed that a pointer-based architecture is advantageous for this task since it can employ task specifics to overcome problems with unknown or rare words, learning to copy those words from the input. We extended the approach by integrating information designed to improve those aspects that need to be generated, especially the appropriate question words. The quantitative evaluation using BLEU scores and an in-depth qualitative evaluation showed that indeed the pointer-based model with additional linguistic features is the best performing system for this task.

References

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2019. On the cross-lingual transferability of monolingual representations. *CoRR*, abs/1910.11856.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Casimiro Pio Carrino, Marta R Costa-jussà, and José AR Fonollosa. 2019. Automatic spanish translation of the squad dataset for multilingual question answering. *arXiv preprint arXiv:1912.05200*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*.

Eldan Cohen and Christopher Beck. 2019. Empirical analysis of beam search performance degradation in neural sequence models. In *International Conference on Machine Learning*, pages 1290–1299.

Sérgio Curto, Ana Cristina Mendes, and Luísa Coheur. 2012. Question generation based on lexico-syntactic patterns learned from the web. *Dialogue & Discourse*, 3(2):147–175.

Kordula De Kuthy, Madeeswaran Kannan, Haemanth Santhi Ponnusamy, and Detmar Meurers. 2020. Towards automatically generating questions under discussion to link information and discourse structure. In *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *International conference on machine learning*, pages 1319–1327. PMLR.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Tobias Kolditz. 2015. Generating questions for German text. Master thesis in computational linguistics, Department of Linguistics, University of Tübingen.

Vishwajeet Kumar, Ganesh Ramakrishnan, and Yuan-Fang Li. 2018. A framework for automatic question generation from text using deep reinforcement learning. *arXiv preprint arXiv:1808.04961*.

Patrick Lewis, Barlas Oğuz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2019. Mlqa: Evaluating cross-lingual extractive question answering. *arXiv preprint arXiv:1910.07475*.

Ming Liu, Rafael A Calvo, and Vasile Rus. 2010. Automatic question generation for literature review writing support. In *International Conference on Intelligent Tutoring Systems*, pages 45–54. Springer.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Liangming Pan, Wenqiang Lei, Tat-Seng Chua, and Min-Yen Kan. 2019. Recent advances in neural question generation. *arXiv preprint arXiv:1905.08949*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Arndt Riester, Lisa Brunetti, and Kordula De Kuthy. 2018. Annotation guidelines for questions under discussion and information structure. In Evangelia Adamou, Katharina Haude, and Martine Vanhove, editors, *Information structure in lesser-described languages: Studies in prosody and syntax*, Studies in Language Companion Series. John Benjamins.

Craige Roberts. 2012. Information structure in discourse: Towards an integrated formal theory of pragmatics. *Semantics and Pragmatics*, 5(6):1–69.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and

position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Leah Velleman and David Beaver. 2016. Question-based models of information structure. In Caroline Féry and Shinichiro Ishihara, editors, *The Oxford Handbook of Information Structure*, pages 86–107. Oxford University Press.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910.