To copy or not to copy?
Text-to-text neural question generation

Tom Hosking

Project Supervisor

Prof Sebastian Riedel

Industry Partner

Dr Guillaume Bouchard, Bloomsbury AI
`http://bloomsbury.ai`

Department of Computer Science

University College London (UCL)

September 2018

I, Tom Hosking, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Asking questions is crucial to human understanding, enabling us to gain deeper insights into the world around us and close gaps in our knowledge. It represents an important and challenging task in the field of natural language processing.

In this study, we implement and train a text-to-text neural question generation model, and show that it is able to produce coherent questions.

We investigate the pointer network, a component used to generate questions that include previously unseen words by copying from the input, and show that careful consideration of when the model should copy can lead to results competitive with the current state of the art.

We experiment with training the model directly on rewards designed to measure question quality, and show that although the models are able to attain higher scores, this does not result in better questions.

Finally, we perform a human evaluation of our models, showing significant improvements over the non-neural baseline.

Acknowledgements

I would like to express my deep gratitude to the following people who have made this dissertation both possible and enjoyable.

Prof Sebastian Riedel for his supervision and support.

Dr Guillaume Bouchard for providing the opportunity to work in industry.

The Bloomsbury AI team for their guidance and advice.

Eric Yuan for his explanations and clarifications.

Alice, Louise, Gerallt, James, Rodney and Lorraine for their time as evaluation workers.

Contents

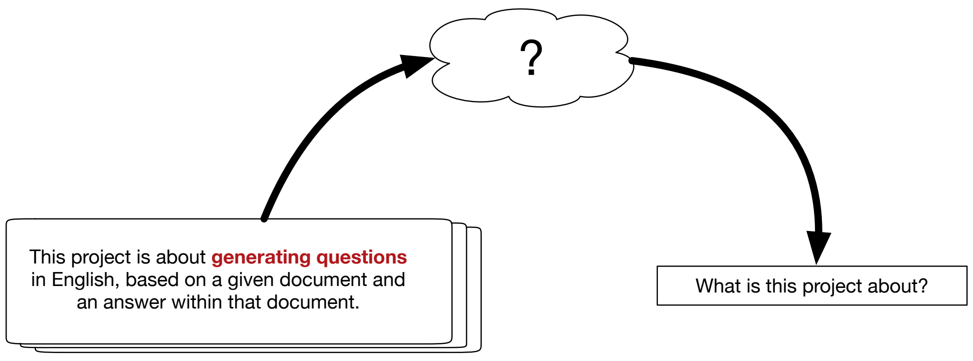
Chapter 1

Introduction

1.1 Motivation

Despite the advent of information technology and the ability to store large amounts of data in carefully structured databases, a large amount of human knowledge is still recorded and communicated via natural language. Scientific papers, legal documents and medical records represent valuable sources of data, and there is significant interest in improving access to the knowledge they contain.

Posing questions about a document in natural language is a crucial aspect of this effort, enabling machines to ask clarification questions (Saeidi et al. 2018), become more robust to queries (A. W. Yu et al. 2018), and to act as automatic tutors (Heilman and Smith 2010). Singer and Donlan (1982) found that students who were asked to pose questions on the material they were studying subsequently went on to perform better in tests on that material.


?
This project is about **generating questions** in English, based on a given document and an answer within that document.
What is this project about?

The ability to ask natural and non-trivial questions involves a number of subtasks, including language modelling, generation and comprehension. A good approach to question generation therefore requires an understanding of these more abstract tasks and is of general academic interest.

While questions often have a unique answer, the inverse is rarely true; there are multiple ways of phrasing a question, and there can be semantically different questions with the same answer. The ability to generate questions therefore provides a mechanism for augmenting existing question answering datasets or automatically annotating new ones, improving the resilience of question answering models.

Generating questions in advance can also help with document recovery, by acting as a form of index. If a new query is the same as (or at least similar to) a generated question about a certain document, then it is likely that this document contains the information that is being sought.

1.2 Bloomsbury AI

Bloomsbury AI was founded in April 2015 with the mission of *scaling access to human expertise*. Their website states:

> Today, much of the world's knowledge is communicated through natural language, through documents or speech. Natural language is hard for computers to understand, so technology hasn't so far been able to give immediate and direct access to this knowledge - even with search it can still take too long, or it can be too difficult, to get the answers you need when you need them. Our mission is to solve this problem - everyone should be able to ask any question and get the answer immediately.

Their primary product, *Cape*, allows a user to upload a document and then query it using natural language. Unlike a search engine that searches for keywords, *Cape* uses machine reading techniques to interpret the question and provide an answer based on the information in the document.

Currently, even small changes in phrasing of a question can result in *Cape* returning different answers. A question generation system could be used to generate varied questions, that could then be used as additional training data to make *Cape* more robust. It would also allow to system to respond to queries with clarification questions - if the query is ambiguous and has multiple valid answers, the system should have the ability to ask the user which of these answers they are looking for.

1.3 Research aims

The primary aim of this project is to implement, evaluate and improve a neural question generation system, and to research a variety of modifications to it. The system should take a text document and an answer within that document as input, and generate a relevant and grammatically correct text question as output.

A machine learning system can be improved by considering three aspects: those of the model design, the training scheme, and the inference method used for sampling from the model. We primarily consider the first two, as well as investigating what “improved" actually means in the context of a question generation system. More concretely, we define the following research questions:

1. *Can we generate natural language questions from a document and an answer, by training a model on existing examples?*
2. *How can we evaluate the quality of the generated questions?*
3. *Is the pointer network a robust approach for generating natural language?*
4. *Does training the model directly on evaluation metrics improve the quality of generated questions?*

1.4 Source code

The source code used for the experiments in this project is available online at `https://github.com/bloomsburyai/question-generation`.

A live demo of our model is also available at `http://qgen.tomhosking.co.uk/`.

1.5 Thesis structure

The rest of this thesis is structured as follows. In Chapter 2 we review existing work in the field, and techniques used for other sequence generation tasks. In Chapter 3 we describe our implementation of a question generation model in detail, and give an initial evaluation of its performance. In Chapter 4 we investigate part of the model architecture in detail. In Chapter 5 we investigate training the model on objectives that are not coupled to the training data. Chapter 6 summarises our results and compares the various approaches. Chapter 7 offers some concluding remarks and suggestions for future work.

1.6 Our contribution

We summarise our contribution as follows:

- We provide a publicly available implementation of a question generation model.
- We propose a split of the SQuAD dataset that allows for true out-of-sample evaluation of our models.
- We show that our model is able to generate questions that are comparable with the state of the art at time of writing.
- We train an additional language model and question answering model to use for model evaluation.
- We provide an interactive demo, allowing a user to query the model on new documents.
- We perform a number of ablations to identify the important components of the model.
- We implement a pointer network component, that may be transferred and used for other work.
- We investigate the behaviour of the pointer network under various configurations, and question a number of assumptions and ambiguities from previous work.
- We perform fine tuning of our models using policy gradient methods, on a range of rewards generated by ancillary neural models.
- We propose a novel modification to a question answering system to directly learn the quality of generated questions from data.

- We use this discriminator model to perform adversarial training of our question generator.
- We evaluate our models using human annotators, comparing to a baseline and to the ground truth.

1.7 Key results

1. *Can we generate natural language questions from a document and an answer, by training a model on existing examples?*

   We show that we are able to generate convincing natural language questions about a document, by training a neural model on crowdsourced examples of questions. While these questions are not as fluent or relevant as those generated by humans, we show that our model outperforms better a non-neural baseline model. We show that reducing the length of document available to the model leads only to a small reduction in quality, with a significant improvement in computational cost. We also show that by removing implicit biases in the training data introduced during encoding, we are able to improve the performance of this model.

2. *How can we evaluate the quality of those questions?*

   We find that, for our purposes, the automatic BLEU metric is a sufficient metric for ranking models based on the quality of the questions they generate. Although the BLEU metric is tied to the ground truth examples, and is a poor estimator of the quality of an individual question, a human evaluation assigns the same ranking to the models as the corpus level BLEU. We also show that a question answering model and a language model may be used to estimate the relevance and fluency of generated questions, but find only a moderate correlation between the human and automatic metrics in this regard.

3. *Is the pointer network a robust approach for generating natural language?*

   The pointer network is a fairly robust method of generating language, that is not overly sensitive to the choice of hyperparameters. We find that a careful consideration of the biases introduced during training can introduce additional freedom to the model, and that this leads to improvements in the quality of generated questions.

4. *Does training the model directly on evaluation metrics improve the quality of generated questions?*

   Policy gradient methods may be used to fine tune the models, increasing their scores under certain metrics to values beyond those achievable only with maximum likelihood learning. However, these increased scores do not lead to improved quality of generated questions when evaluated by human workers.

Chapter 2

Background

In this chapter, we describe some key areas of research related to our study, including work on the question generation task as well as relevant techniques and concepts from other fields.

2.1 Datasets

Over the past few years, a range of machine comprehension datasets have been made available, consisting of triples of documents (or *contexts*), questions and answers. These have generally been used to train question answering (QA) models to infer the answer from the context document and question, but they can also be used for the question generation or *answer questioning* (AQ) task. In the AQ task, the context and answer are provided as inputs, and a model must generate a plausible question as output.

Each of these datasets comes with its own difficulties and biases. Hermann et al. (2015) present a set of cloze questions[1] generated automatically from news stories, while Hill et al. (2015) take a similar approach with childrens stories by removing words from sentences at the end of the context. Y. Yang et al. (2015) released WikiQA, comprising 3047 *natural* questions on Wikipedia articles, but this dataset is considered too small for most deep learning approaches.

TriviaQA from Joshi et al. (2017) includes 95k question-answer pairs that are constructed by quiz enthusiasts without reference to a specific context, and therefore require a model to first find the correct document before pointing to the answer. The MS Marco dataset constructed by

[1]A cloze question takes the form of a statement with one or more words missing - the task is then to determine the word(s) that has been removed.

Nguyen et al. (2016) uses real human queries collected from a search engine, with answers that are not necessarily explicitly present in the context.

Trischler et al. (2016) present NewsQA, 119,633 crowdsourced questions on 12,744 news articles, with answers potentially spanning multiple tokens, and with some questions having no valid answer within the context. The Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al. 2016) comprises a total of 107,785 crowdsourced question-answer pairs, based on 536 distinct documents. Rajpurkar et al. (2018) updated this dataset with SQuAD v2, to include an additional 50,000 adversarial examples - that is, context-question-answer triples that seem plausible at first glance but are actually invalid.

Dataset	*Natural questions*	*Large Scale*	*Self contained*	*Valid answer in context*
Hermann et al. (2015)	-	-	✓	✓
Hill et al. (2015)	-	-	✓	✓
WikiQA	✓	-	✓	✓
TriviaQA	✓	✓	-	✓
MSMarco	✓	✓	-	-
SQuAD v2	✓	✓	✓	-
SQuAD	✓	✓	✓	✓

Table 2.1: QA datasets and their properties.

Since our task is to invert the usual question answering task and generate questions conditioned on a context and answer, we require the answers in the data to both *exist* and be *valid*, ruling out the use of NewsQA and SQuAD v2. We also do not wish to require any knowledge beyond that contained within the context document to generate the questions, and so TriviaQA and MS Marco would be poor choices. We therefore use SQuAD v1 (without the adversarial examples) as our dataset for training and evaluating our model.

2.1.1 SQuAD

These desirable properties that led us to choose SQuAD are not always true. In rare cases, the answer words do not actually appear in the context as words, but rather as sequences of characters within other words, as shown in Table 2.3. In other cases, as shown in Table 2.2, some external knowledge is required; although the answer is contained within the context, *interpreting* the question correctly requires knowing that the team is known as the “Denver Broncos" and that “Denver" in the question is a reference to “Broncos" in the context.

Context
[...] the *broncos* finished the regular season with a 12-–4 record , and denied the **new england patriots** a chance to defend their title from super bowl xlix by defeating them 20-–18 in the afc championship game . [...]
Ground Truth Question
who did denver beat in the afc championship ?

Table 2.2: Example of a difficult ground truth question. The word “denver" does not appear in the context, and so cannot be copied. Some form of external knowledge would be required here to relate it to “broncos" which does appear.

Figure 2.1 shows the distribution of types of question in the SQuAD training set, as determined by the question word or *interrogative* used. The category “other" contains questions such as the examples in Table 2.4 that are binary yes/no questions, or indeed questions that are tasks rather than actual questions.

Context
wes**two**od one will carry the game throughout north america , with kevin harlan as play-by-play announcer, boomer esiason and dan fouts as color analysts , and james lofton and mark malone as sideline reporters . jim gray will anchor the pre-game and halftime coverage .
Ground Truth Question
how many color analysts were involved with super bowl 50 ?
Answer
two

Table 2.3: Example of a sub-word answer - “two" does not itself appear as a token in the context, but the characters are present within the token “westwood".

Table 2.4 shows an example of the noise that is introduced through the crowdsourcing process - the word “how" has been misspelled, making this example hard for an automated system to interpret.

Misspelled words:	hoe did everyone learn that beyonce performed for kaddafi ?
Binary interrogative:	does montana have a sales tax ?
No question mark:	name four sound profiles that would result in bass distortion on pre-2007 ipods .

Table 2.4: Example of quirks within SQuAD questions.

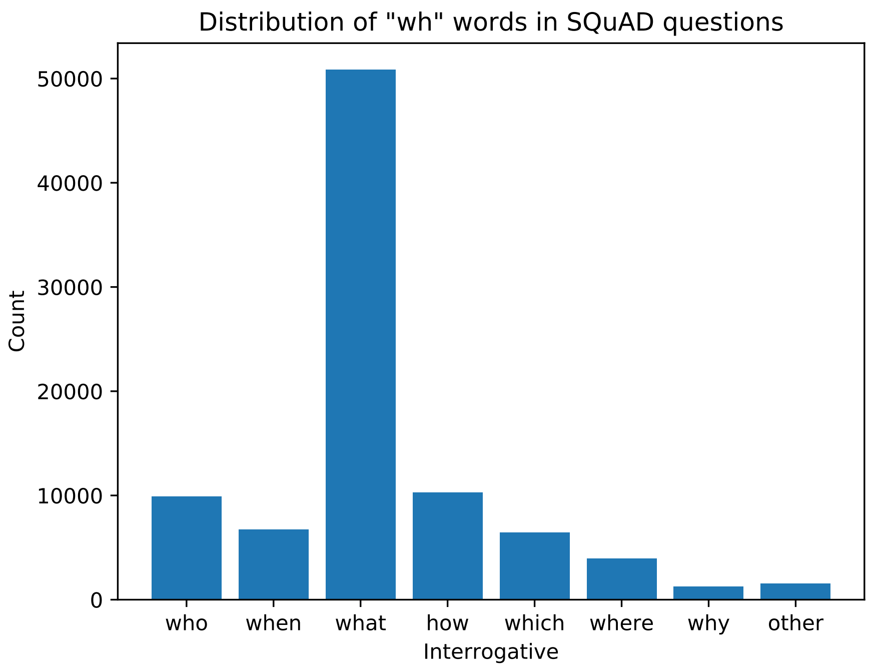


Figure 2.1: Distribution of interrogatives in SQuAD training set.

2.1.2 Validation split

In order to protect the integrity of the SQuAD test set, it is hidden from the public, and submissions of models to the QA leaderboard must be run by the curators. This is good for maintaining a rigorous QA task, but means that there is not a natural test set available for the question generation task.

Yuan et al. (2017) evaluate their AQ model over the full SQuAD development set, and it is not clear what data they use for their model selection or early stopping.

Zhou et al. (2018) create a test set by randomly splitting over all context-question-answer triples in the development set to create a new development and test set. The official development set includes multiple crowdsourced answers for each context-question pair, to allow for possible ambiguities in the questions: QA models are evaluated by taking the maximum score across these possible answers. However, including these additional answers when enumerating all possible triples means that the same question may appear both in the development and test sets. The test set proposed by Zhou et al. (2018) is therefore not out-of-sample.

We therefore propose our own split, in keeping with the method used by Rajpurkar et al. (2016). We split the set of *documents* in the official development set randomly in half to give our development and test sets. We use all questions for each document, and use only one crowdsourced answer for each question. Although this results in an different number of triples in the development and test sets, questions are no longer repeated within splits and are not duplicated across both development and test split. The documents in each split are also distinct and cover different domains, making our development and test splits a better indicator of out-of-sample performance.

Our development set contains 4961 triples, and our test set contains 5609 triples. We publish this split in the standard SQuAD data format online[2].

Gao et al. (2018) concurrently proposed a similar approach, but have not released the split they used.

2.2 Artificial neural networks for NLP

The advent of artificial neural networks has facilitated the practice of *representation learning*, shifting the research focus from feature engineering, to designing and training neural architectures that can learn those features directly from data. In this section we describe some techniques that are particularly relevant to the task of generating questions and to working with natural language in general.

[2]`https://github.com/bloomsburyai/question-generation/tree/master/data`

2.2.1 Word embeddings

A naive approach to encoding a sequence of words would be to define a limited (but arbitrarily large) ordered set of V distinct possible tokens (i.e. a vocabulary), and to represent a given token as a vector $\mathbf{x} \in \mathbb{R}^V$ where $x_i = 1$ for the index i of the word in the set, and 0 otherwise. The set of possible tokens therefore forms an orthogonal basis. While this approach, known as a *one hot* representation, is successful in that it encodes a string of characters into a real valued vector, the assumption that all words are orthogonal to each other is problematic. For example, “cat" is more similar to “dog" than it is to “ambition", and the encoding of the data should reflect this.

Word embeddings instead seek to find a lower dimensional space $\mathbb{R}^d$ with $d \ll V$ and embed each word in the vocabulary as a point in this space. These embeddings should capture the observation that some words are similar to others, and thus contain some level of meaning.

Converting a sequence of tokens to embeddings requires that we define a fixed size vocabulary that a model will be able to recognise. This vocabulary normally also includes a special *out-of-vocabulary* (OOV) token, and any input tokens not in the vocabulary are mapped to OOV.

Embedding vectors are a natural way to encode sequences of words as inputs to an artificial neural network, and indeed can be learnt along with the other model parameters.

Y. Bengio et al. (2000) learnt embeddings to predict the next word in a sequence using a language model based on a multilayer perceptron, while more recent work has focussed on finding task-independent embeddings. Following the idea popularised by Firth (1957) that “a word is characterised by the company it keeps", *word2vec* (Mikolov et al. 2013b) is a method for learning embeddings for words based on the context in which they appear. The Stanford Global Vectors (GloVe) (Pennington et al. 2014) are a learned set of embeddings such that the dot product between two embedding vectors approaches the logarithm of the co-occurrence probability of those words.

Where relevant, the embeddings in the models described in this paper are initialised using publicly available GloVe vectors pretrained on *Wikipedia 2014 + Gigaword 5* corpora, with a total of six billion tokens and a vocabulary size of 400,000. This initialisation can be viewed as

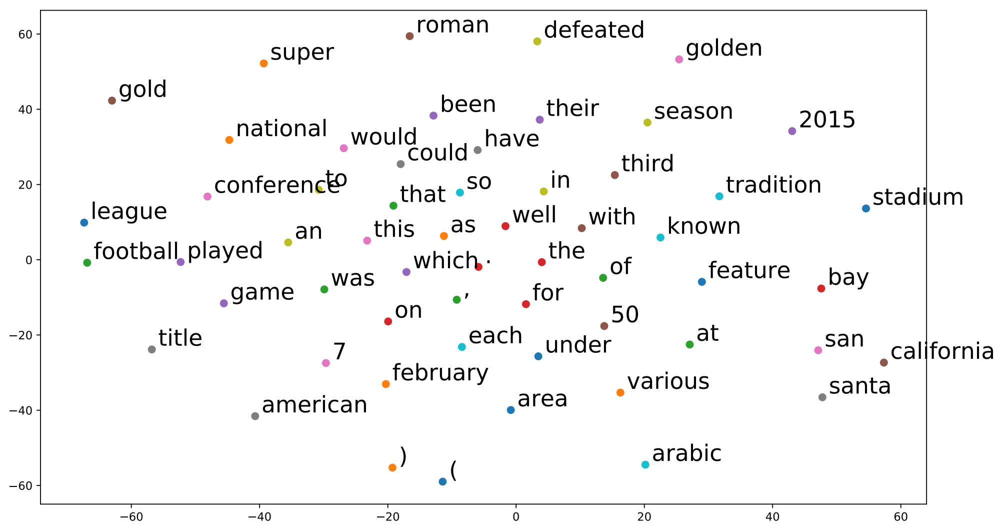


Figure 2.2: Visualisation of GloVe embeddings, showing that similar words are embedded close to each other

a form of *transfer learning*, where a model trained on one task is used to speed up training on another (related) task by incorporating knowledge external to the training data.

Recent work by Peters et al. (2018) has considered the goal of generating context-specific embeddings using a bidirectional language model known as ELMo, reflecting the observation that the meaning of a word can change depending on its context. For example, “play" may be a verb or a noun, and this ambiguity may be resolved by considering the usage of the word.

2.2.2 Sequence models

An embedded sequence consists of a list of vectors, which is generally of variable length. This variability means standard fully connected layers are not appropriate for processing language data, and indeed for longer sequences would require a prohibitively large number of parameters. Current approaches generally make use of either *recurrent* or *convolutional* methods.

Recurrent neural networks (RNNs) were first described by Little (1974), and popularised by Hopfield (1982). An extension of the standard neural network, they introduce a connection forward in time between nodes within a layer, by conditioning the output at each time step both on the input and on a hidden state passed from the previous time step. Figure 2.3 shows a representation of a RNN unrolled in time. Training the parameters of a RNN can be achieved with back-propagation through time, but suffers from the vanishing or exploding gradients problem (Y. Bengio et al. 1994; Hochreiter et al. 2001) where the magnitude of the errors grows or shrinks due to compounding. Hochreiter and Schmidhuber (1997) introduced the Long-Short Term Memory (LSTM) cell, and Cho et al. (2014) proposed the Gated Recurrent Unit (GRU), both designed to avoid vanishing and exploding gradients in deep recurrent models over long sequences.

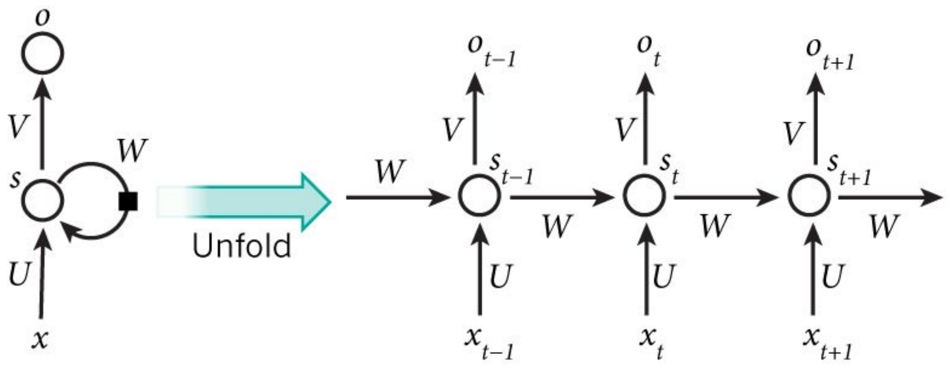


Figure 2.3: Visualisation of the unrolling of a recurrent neural network through time. From Y. A. LeCun et al. (2015)

An alternative approach is to treat the embedded sequence as a $2D$ array of data, and make use of convolutional techniques. Described in detail in Krizhevsky et al. (2012), convolutional neural networks (CNNs) were introduced by Y. LeCun et al. (1990) and led to major advances in the field of computer vision (Deng et al. 2009; He et al. 2015). Briefly, the input data is convolved at each layer with a learned kernel. While RNNs must calculate the output at one time step before being able to calculate the next, CNNs can be calculated in parallel and so are significantly more efficient to run on GPUs. A. W. Yu et al. (2018) made use of this training speedup to train a QA model on a much larger, augmented SQuAD dataset, reaching state of the art performance at the time. Gehring et al. (2017) applied CNNs to the NMT task, and showed that they can also be used for sequence generation. While RNNs remain the method of choice in NLP, Bai et al. (2018) argue that CNNs should be preferred.

The transformer architecture (Vaswani et al. 2017), that uses self-attention to process sequential data, has shown promising results with reductions in computational cost.

2.2.3 Sequence generation

As well as taking a sequence as input, we also consider the task of generating a new sequence as output. This can be viewed as a sequential decision process, where a model calculates a distribution over output tokens at each time step, conditioned on some internal state and the output up to that point. Sampling from this distribution then produces a concrete output sequence. For example, unconditioned sentences may be generated by simply sampling from a language model (see Section 2.6.1). Sutskever et al. (2014) proposed an encoder-decoder framework known as *Seq2Seq* for reading an input sequence, generating a fixed length representation, and then *decoding* this representation to form an output sequence. Essentially, the input is fed into an encoder RNN, and the final hidden state used to initialise a second decoder RNN, from which output tokens are then sampled. This has been used to great effect in the field of neural machine translation (NMT).

In the context of learning alignments between input and output tokens in NMT, Bahdanau et al. (2014) introduced the concept of *attention*. An attention mechanism generates a distribution over input time indices for each output step, allowing a sequence decoder to 'attend' over different parts of the input sequence when decoding. This helped to improve NMT performance on longer sequences, by providing different a path for information to flow around the fixed-length bottleneck.

A common problem in NMT (and indeed sequence generation in general) is that of *unknown words*. Although the number of words in a dictionary is finite, in practice sentences may contain a wide range of names, places or other *named entities* as well as numbers, abbreviations, slang etc. The fixed vocabulary the model must operate over therefore becomes impractically large if OOV tokens are to be avoided. Jean et al. (2014) proposed a technique for making the use very large vocabularies practically feasible, but it is not easy to learn good embeddings for the very rare words in this vocabulary.

Pointer networks were introduced by Vinyals et al. (2015) and proposed as a possible solution to the unknown words problem by Gulcehre et al. (2016). At each time step the model produces a distribution over both the shortlist vocabulary and the tokens in the input sequence. This method allows the model to copy tokens from the input sequence, even if those tokens have never been encountered before. Gu et al. (2016a) proposed a similar architecture called CopyNet that calculates logits instead of probabilities over each vocabulary, which are then normalised as a whole using the softmax function.

2.2.4 Training

When training sequence generation models, it is common to use the technique of *teacher forcing* (Williams and Zipser 1989). Instead of using the previous output token as input at the next step, the ground truth is used. This trains the network on a one-step-ahead prediction task, and improves the speed of convergence.

Since the number of parameters in neural models is often very large, and models can theoretically have the capacity to simply remember the training data, some form of regularisation is usually required to ensure good generalisation. We make use of dropout (Srivastava et al. 2014), whereby a random set of activations in the network are zeroed out for each sample. This prevents the model from being overly reliant on a small set of connections and helps prevent feature co-adaptation. We also make use of variational dropout (Gal and Ghahramani 2015) for sequence models, where the same set of connections are disabled for each time step in the sequence.

We also use early stopping (Morgan and Bourlard 1989); during training, the performance of the model is evaluated on a development set, and training is stopped after this performance stops improving[3]. This helps avoid overfitting to the training set.

To avoid the exploding gradient problem (Y. Bengio et al. 1994; Hochreiter et al. 2001), we follow Pascanu et al. (2012) and apply gradient clipping based on a maximum global norm of five.

2.3 Improving output quality

2.3.1 Exposure bias

The process of teacher forcing described earlier leads to a discrepancy between training and inference. During training, the correct token is fed as input at each step, while during inference

[3]A “patience" parameter is used to avoid stopping too early - the performance must not improve for this number of evaluations before training is stopped.

the previous output token is used. This leads to *exposure bias* (Ranzato et al. 2015) where the model is not exposed to its own mistakes during training, allowing errors to quickly compound during inference. Specifically, the model does not learn how to distribute probability mass among “second best" examples, even though some may be valid and some may be completely incoherent. S. Bengio et al. (2015) proposed scheduled sampling as a possible solution; the input token is selected probabilistically from either the previous output or the ground truth according to a schedule. However, Huszár (2015) showed that scheduled sampling has weak theoretical justification and can lead to inconsistent learning.

Early experiments showed a propensity for our model to produce short questions. Without some form of normalisation, the overall log-likelihood of longer sequences is lower, and so they are not generated. Y. Wu et al. (2016) propose a normalisation method given in Equation 2.1 that allows the scores to be skewed towards longer sequences, giving a hyperparameter that may be adjusted to encourage the model to produce longer questions.

$$lp(Y) = \frac{(5 + |Y|)^{\alpha}}{(5 + 1)^{\alpha}} \tag{2.1}$$

We found empirically that a value of 0.12 gave good results.

2.3.2 Beam search

Greedily sampling from a sequence generation model (whether a Seq2Seq type model or a language model), by taking the most likely output at each step, does not necessarily lead to the sequence that has the overall highest likelihood. Ideally, all possible sequences would be generated, and the sequence with the highest likelihood retained; in practice, this quickly becomes unfeasible as the length of sequence increases. Beam search decoding (Graves 2012) attempts to balance these concerns by maintain a fixed length k *beam* of sequence hypotheses. At each step, the top k possible extensions of each hypothesis are sampled, and the total likelihood of the extended sequences updated, giving k^2 new hypotheses. This list is then cut down to the top k most probable hypotheses, and the process repeated until decoding terminates. Following Yuan et al. (2017), we set $k = 32$ for our experiments.

Beam search involves a *top-k-argmax* operation that selects the top k scores at each time step. This process is not differentiable[4] (Goyal et al. 2017), meaning that backpropagation may not be used to update the parameters of the model based on a loss calculated on a sequence sampled with beam search.

It is worth considering why sampling is necessary at all: why not keep the full distribution at each step and allow the model to use this additional information? A useful analogy is to consider a control system that steers a car; if an obstacle appears directly in front of the car, the safe path requires steering to the left or to the right. Both of these choices would add similar distance to the journey, so the control policy might assign roughly equal probability to turning left or right. Then, the *mean* policy is in fact to perform no steering at all, which would lead to a collision with the object. A good decision made is only when the probabilistic policy is sampled and becomes discrete.

A parallel with natural language might be a sentence with multiple adjectives. In English, adjective order is often not important, for example “the red fast car" and “the fast red car" are both valid and equivalent phrases. Unless the choice of the second adjective is conditioned on the discrete choice of the first, the invalid phrase “the red red car" could feasibly be generated.

2.3.3 Policy gradient methods

All the models described so far are trained using maximum likelihood; that is, the parameters are optimised to maximise the probability of reproducing the training data. In the case of question generation, the particular choice of wording selected by the crowdworkers is almost certainly not the only way of phrasing that question, and indeed there are often multiple distinct questions with the same answer. The model should not be adversely judged for generating a question that is valid and makes sense, but happens to be different to the question generated by the crowdworkers. In the absence of multiple ground truth questions, it is therefore desirable to partially decouple the model from the training data, and attempt to optimise the performance metrics directly. Norouzi et al. (2016) confirmed that taking such a reward into account (by weighted sampling) improved the quality of output. Using rewards calculated directly from generated questions would also help with the problem of exposure bias, as the model may

[4]Not differentiable in this context means that the outputs are not differentiable with respect to the inputs or to the parameters.

then explore the search space more widely, and learn how to distribute probability mass among sequences that do not appear in the training set.

Almost all metrics on a generated question require a complete and concrete example: it is hard to know whether a question is grammatically correct or meaningful if it is incomplete. Beam search is a crucial component in generating meaningful sequences and therefore should be used, but as discussed it is not a differentiable operation. If the sequence generation task is treated as a sequential decision process, where the output distribution is considered as a policy and the tokens are considered as actions, policy gradient techniques may be used to update this policy to maximise external non-differentiable rewards. The REINFORCE algorithm (Williams 1992), given in equation 2.2, relates the gradient of the expected reward under a policy to the expectation of a gradient, which can then be approximated by sampling with standard minibatch gradient descent.

$$\nabla_\theta \mathbb{E}_\pi[R(s,a)] = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) R(s,a)] \tag{2.2}$$

Policy gradient methods have been used to improve the output of dialogue generation systems (Li et al. 2016), sentence simplification models (Zhang and Lapata 2017), chatbots (Kandasamy et al. 2017) and to optimise sequence generation models directly on performance metrics (Ranzato et al. 2015). They have also been used in the context of NMT (L. Wu et al. 2017; Zhen Yang et al. 2017), although the performance improvements have not been sufficiently large for the technique to become standard practice.

If the reward is generated by a neural model, this model can be updated during training. Generative adversarial networks (GANs) (Goodfellow et al. 2014) set up a minimax competition between a generator and a discriminator; the discriminator must learn to classify samples as either generated or real (in the sense that they came from the training data), and the generator must learn to fool the discriminator by generating samples that are similar to training data.

L. Yu et al. (2016) proposed a framework known as SeqGAN, that uses REINFORCE to train a sequence generator alongside an adversarial discriminator. Since the reward is generally a single score over the whole sequence, the signal can be quite noisy and sparse: the generator is not told *why* the sequence was good or bad. Che et al. (2017) considered a modified training objective to improve stability of learning based on importance sampling. Both approaches use Monte Carlo (MC) search to update the policy at each step; a partially generated sequence is updated according to the mean reward from a sampled set of full sequences that could be subsequently generated.

The adversarial approach was used to improve a NMT system (Zhen Yang et al. 2017) while Tuan and Lee (2018) apply it to artificial conditional sequence generation tasks. Fan et al. (2018) use an adversarial approach to generate questions conditioned on images, rather than text documents.

2.4 Baseline models

For our baseline model, we use the rule-based PCFG-Trans system (Heilman 2011), adapted by Zhou et al. (2018) for use with SQuAD context-answer input pairs and available publicly[5]. This is a non-neural model that extracts multiple factual sentences from the context, converts them into candidate questions, and automatically ranks them so that better questions are more likely to be selected.

2.5 Evaluation metrics

In order to evaluate the performance of a question generation model, we make use of a number of automatic metrics. For all context-answer pairs from SQuAD that will be used to generate questions, there is a human generated question available for comparison.

2.5.1 Similarity to ground truth

In the context of sequential data, accuracy measures the fraction of tokens that are the same in the predicted and ground truth or *gold* sequences at each time step. Accuracy is therefore sensitive to insertions or deletions - taking a ground truth sequence “what year was Tom born ? <Pad>", the candidate “in what year was Tom born ?" appears very similar to human eyes, but would be assigned an accuracy of 0, since the candidate never has the same word at the same time step as the ground truth.

[5]`https://res.qyzhou.me/`

The definition of F1 score for sequences as given by Rajpurkar et al. (2016) is more flexible - each sequence is treated as a bag of words, and the standard F1 score[6] is then calculated for these unordered sets. For the example above, the F1 score is 0.86, reflecting the word overlap between the two sentences. However, the treatment of the sequence as unordered is problematic: the sentence “born Tom year ? what in was" is assigned the same F1, but is clearly nonsense.

BLEU (Papineni et al. 2002) seeks to improve on F1 by considering n-grams[7] instead of single words, and returning a modified form of the geometric mean of the n-gram precisions up to some limit (usually 4). This goes some way towards resolving the issues with F1; the first candidate now scores 0.81 and the nonsense sentence scores 0.

Since the BLEU score is the geometric mean of up to n-gram precisions, if any of those precisions are 0 then the whole sequence is assigned a score of 0. For example, the sentence “in what year is Tom born ?" is very close in semantic meaning to the ground truth, but since there are no 4-grams overlapping between the two, the BLEU score is 0. The instance level BLEU is therefore a poor indicator of the quality of an individual question. Papineni et al. (2002) acknowledge this, and instead suggest evaluating a corpus as a whole by calculating the precision for each order of n-gram over the whole set of predictions. The corpus-level BLEU is then the geometric mean of these corpus-level precisions. In this report, BLEU score in general refers to a corpus-level score.

BLEU has been shown by (Chaganty et al. 2018) to be a biased estimator that correlates poorly with human evaluation of language. However, no preferred alternative has yet been proposed, and so we report BLEU scores as our primary similarity metric.

METEOR (Banerjee and Lavie 2005) attempts to provide a more robust metric by normalising the sequences for synonyms and stemming. This in turn could introduce a bias from the choice of stemming method, and indeed is no longer computationally fast or simple to implement, so we do not report it.

White et al. (2017, 2018) found that it is often possible to reconstruct short sentences from only the sum of the embeddings of the words they contain, and indeed Mikolov et al. (2013a)

[6]The F1 is given by the harmonic mean of the precision and recall.
[7]A n-gram is a sequence of words of length n.

investigated the use of word embeddings as a method for comparing phrase meanings. While comparing sums of word embeddings (SOWE) between two sequences will not account for word order, unpublished work by White et al. shows that it can be used successfully to predict human rankings of generated sequences, and therefore acts as a measure of quality of content. We experiment with calculating the cosine similarity given by

$$\text{SOWE} = \frac{\mathbf{v_g}.\mathbf{v_p}}{\|\mathbf{v_g}\|\,\|\mathbf{v_p}\|}, \tag{2.3}$$

where

$$\mathbf{v_p} = \sum_t^{|\hat{Y}|} \mathbf{e}_t, \tag{2.4}$$

$$\mathbf{v_g} = \sum_t^{|Y|} \mathbf{e}_t, \tag{2.5}$$

and $\mathbf{e}_t$ is the embedding vector of the word at position t in each sequence, to measure to similarity in content between generated questions and ground truth.

2.5.2 Question quality

As discussed previously, the training data does not represent the only valid question that could be generated. We therefore also seek to measure the quality of the questions directly.

We trained a LSTM language model on the questions in SQuAD, and report the perplexity (defined in Equation 2.6) of generated questions under this language model, as an indication of the fluency or quality of language of the generated questions. The language model is described in detail in Section 5.1.1.

$$\text{PPL} = 2^{-\sum\limits_t p(x_t)\log_2 p(x_t)} \tag{2.6}$$

As well as being valid language, generated questions should also be relevant to the document and consistent with the answer. We trained a question answering model on the SQuAD training set, and calculate the F1 score attained by this model when it attempts to answer the generated questions. In some sense this acts as a reconstruction loss - we should be able to recover the same answer that was used to generate the question. The QA model used is described in detail in Section 5.1.2.

Finally, based on our work in Chapter 5 we experiment with using a discriminator to assess the quality of questions, where the discriminator gives a probability that a given sequence was generated or came from the ground truth dataset. Such a model may in theory learn to assess the quality of generated questions directly from data.

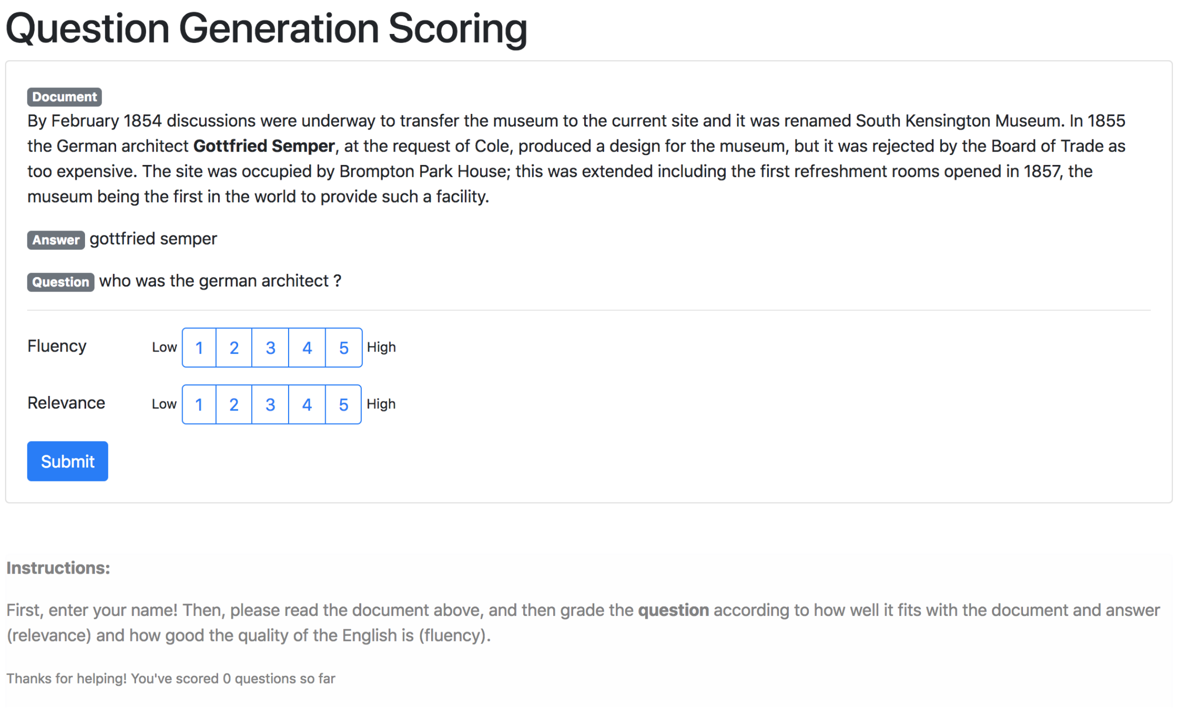
2.5.3 Human evaluation

Ultimately, the best judge of quality of a generated question is a human. As shown by Chaganty et al. (2018), automated metrics are not a sufficient replacement for human judgement. For the question generation task, the problem is further compounded since there are multiple valid questions of which the ground truth is only one example.

Since manual evaluation of question quality is a time consuming task, we set up a human evaluation to collect scores for only a limited set of models, on a limited set of generated questions. In order to be able to compare scores from different workers on the same example and same model, while also getting good coverage across the different models, we selected a random subset of examples from our test set, and generated questions for each of them with every model. The task was then presented to workers as follows: the context and answer were picked sequentially from the subset of test set examples, and the model used to generate the question was selected at random. We included the baseline and the ground truth in the list of possible sources. Each worker was therefore presented with every context-answer pair only once. A screenshot of the annotation interface presented to workers is shown in Figure 2.4.

We follow the standard approach in evaluating machine translation systems (Koehn and Monz 2006) as used for AQ by Du and Cardie (2018), that also inspired the choice of rewards made by Yuan et al. (2017) for their model. We ask workers to rate each generated question between 1 (poor) and 5 (good) on two separate criteria: the quality of the language used, and how well the content of the question is related to the context document and answer. We refer to these measures as the *fluency* and *relevance*[8] of the questions.

[8]The standard term used in machine translation is adequacy, i.e. how adequately does the translation convey the meaning of the original text. This term is less well suited to describing whether a question is specifically


Question Generation Scoring

By February 1854 discussions were underway to transfer the museum to the current site and it was renamed South Kensington Museum. In 1855 the German architect **Gottfried Semper**, at the request of Cole, produced a design for the museum, but it was rejected by the Board of Trade as too expensive. The site was occupied by Brompton Park House; this was extended including the first refreshment rooms opened in 1857, the museum being the first in the world to provide such a facility.

gottfried semper

who was the german architect ?

Instructions:

First, enter your name! Then, please read the document above, and then grade the **question** according to how well it fits with the document and answer (relevance) and how good the quality of the English is (fluency).

Thanks for helping! You've scored 0 questions so far

Figure 2.4: Screenshot of the scoring interface used to collect human evaluations

2.6 Related tasks

2.6.1 Language modelling

Language modelling refers to the task of predicting the next word in a sequence, conditional on all words in the sequence so far. A question generation system could be viewed as a language model that is also conditioned on the context-answer pair currently under consideration.

Non-neural methods generally involve keeping track of pseudo-counts of words that follow each distinct n-gram, and using some form of smoothing to deal with the fact that many of the possible n-grams will not be seen at training time. Kneser-Ney smoothing (Ney et al. 1994) describes an algorithm that recursively calculates the weighted average of the probability of a given token occurring after the most recent n tokens in the sequence and the Kneser-Ney result for $n - 1$ tokens.

More recently, it has become common to use a recurrent architecture with LSTM or GRU cells, where a sequence of tokens is represented as a sequence of embeddings, then fed as input to a RNN (Sundermeyer et al. 2012). This helps solve the problem of unknown n-grams, as the

related to the content of the context and answer.

output of the cell at each step is only conditioned on the input at that step and the hidden state carried over from the previous step.

2.6.2 Question answering

Question answering (QA) involves a model taking a context document and question as inputs, and predicting an answer based on those inputs. In the case of SQuAD, this answer is known to be contained within the context, and so it is sufficient to predict the location of the answer. Current QA models draw on many of the approaches discussed so far, and we briefly describe only those particularly relevant to our work.

In their paper on question generation, Yuan et al. (2017) use a multi-perspective context matching (MPCM) question answering model from Z. Wang et al. (2016) as a reward mechanism and evaluation metric. This model has a similar architecture to the more widely known bi-directional attention flow (BiDAF) (Seo et al. 2016) model, where the key component calculates a similarity matrix between the tokens in the input question and context and uses this information to generate a query-aware representation of the context that can then be used to locate the answer.

QANet (A. W. Yu et al. 2018) is a convolutional question answering model, with one of the highest scores on SQuAD at time of writing. Although convolutional rather than recurrent, the model takes a similar form, with question and context encoder layers, and an attention mechanism to combine the information from these two inputs.

2.6.3 NMT and summarisation

Neural machine translation (NMT) has led to a number of advances in sequence modelling, discussed in earlier sections (Bahdanau et al. 2014; Gehring et al. 2017; Gulcehre et al. 2016; Sutskever et al. 2014), but for NMT the input and output sequences often have comparable lengths (Shi et al. 2016). This means that the attention weights correspond closely to per-token alignments, and the task becomes closer to a context-aware per-token transformation. Question generation requires identifying the important sections of the context before reordering some phrases and constructing some new ones, and so poses somewhat different challenges.

Shi et al. (2016) showed that NMT models keep track of the length of the decoded sequence, and that for western languages the correct output sequence length is comparable to

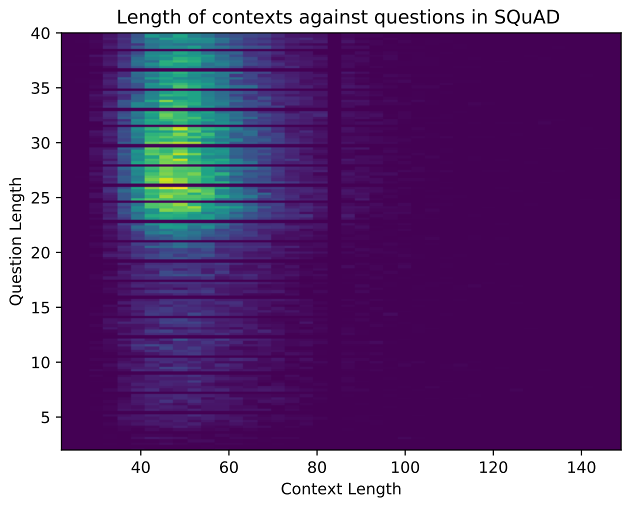


Figure 2.5: Visualisation of context and question lengths in SQuAD. The correlation coefficient is 0.0097.

the input length. For question generation, there is no particular reason that the length of the input context or answer should have any bearing on the length of the question, and as shown in Figure 2.5 their lengths are not correlated.

Summarisation involves taking a context document as input and generating a summary that should be considerably shorter, and is therefore more similar to the task of question generation. Cheng and Lapata (2016) propose an extractive[9] neural summarisation framework that makes use of a Seq2Seq architecture with an attention mechanism, and Gu et al. (2016b) extend this by adding a pointer network. Nallapati et al. (2016a,b) present an abstractive model, and See et al. (2017) again extend these approaches by adding a pointer network.

2.7 Question generation

Early systems for generating questions were generally based around the use of some sort of templating or rule-based reordering of the context tokens. By identifying the role of the answer phrase in the document and the type of concept it represents, the correct interrogative (e.g. “when") and relevant contextual information can be extracted and rearranged to form a question.

[9]Extractive models use words and phrases in the context to generate a summary, whereas abstractive models aim to paraphrase.

Agarwal et al. (2011) split the task first into content selection, then question formation by transforming the relevant phrases. Heilman and Smith (2010) and Heilman (2011) also transform declaratives into candidate questions, then rank these candidates to select the best output. Other approaches have used syntactic parsing (Ali et al. 2010; Danon and Last 2017) or template based methods (Chali and Golestanirad 2016; Labutov et al. 2015; Mazidi and Nielsen 2014; Popowich and Winne 2013) to construct the output. The context document is parsed into a set of relations or ontologies, the type of the relation is used to select a template, and the template completed using the nodes of the relation.

AQ systems could be used to provide an automated method for annotating data and augmenting datasets for training QA models, and Tang et al. (2017) and T. Wang et al. (2017) approach the task with this in mind. They generate questions using a Seq2Seq model, but primarily focus on the resulting improvement to the QA model and do not directly assess the quality of their generated questions. Zhilin Yang et al. (2017) take a similar approach, using an AQ model to facilitate semi-supervised training of a QA model on a range of different domains.

Similar to earlier template based approaches, neural techniques have been used to generate questions from entities and relations in a knowledge base (Indurthi et al. 2017; Serban et al. 2016), but these require knowledge of the relations in advance and do not work from the raw textual input.

The AQ task has also been approached by using only the document to generate questions, without conditioning on a specific answer. Subramanian et al. (2017) used named entity recognition to identify key phrases in the context before feeding this reduced input to a Seq2Seq model. They report an improved rating by human evaluators compared to their baseline model (Heilman and Smith 2010), but do not give any automated evaluation metrics for the generated questions. Kumar et al. (2018a) use a similar two-stage approach, adding attention and a pointer network to the decoder. Kumar et al. (2018b) further update this model by performing secondary policy gradient training, using BLEU and other automatic metrics as the rewards.

Du et al. (2017) use a Seq2Seq based model to generate questions from context-answer pairs, and build on this work by preprocessing the context to resolve coreferences[10] and adding

[10]Coreferences are different words in a document that all refer to the same entity. For example, “she" is generally used as a pointer to refer to a person that has been named earlier in the text.

a pointer network (Du and Cardie 2018). Similarly, Zhou et al. (2018) use a part-of-speech tagger to augment the embedding vectors. Both Du and Cardie (2018) and Zhou et al. (2018) perform a human evaluation of their models, and show significant improvement over their baseline (Heilman 2011). Song et al. (2018) use a modified context encoder based on multi-perspective context matching (Z. Wang et al. 2016), similar to cross attention. Bahuleyan et al. (2017) used a variational encoder to generate multiple questions from a single context sentence.

Yuan et al. (2017) similarly describe a Seq2Seq model with attention and a pointer network, with an additional encoding layer for the answer. They also describe a method for further tuning their model on a language model and question answering reward objective using policy gradient, but unfortunately do not perform any human evaluation.

Gao et al. (2018) propose splitting the training data by the difficulty of question, and including this difficulty as part of the conditioning on the decoder. They find that this helps to improve the BLEU scores of their generated questions, although they do not perform a human evaluation.

In summary, most recent work has centred around a Seq2Seq model with attention and a pointer network, with various choices of input encoding or feature augmentation. We proceed by replicating such a model, and investigating some of the modifications proposed by Yuan et al. (2017).

Chapter 3

Generating questions

Yuan et al. (2017) describe a question generation model with some promising results; however, they were not able to release their implementation of this model to the public. We therefore begin by implementing their model and evaluating its performance.

3.1 Preprocessing pipeline

The SQuAD dataset is provided as a set of natural language context-question-answer triples (D, Q, A). These strings are tokenised into sequences $(\mathbf{D}, \mathbf{Q}, \mathbf{A})$ using the NLTK punkt tokeniser (Loper and Bird 2002), and all tokens transformed to lowercase[1].

3.2 Model architecture

The question generation model from Yuan et al. (2017) is based around a Seq2Seq architecture (Sutskever et al. 2014), with attention (Bahdanau et al. 2014) and a pointer network (Gulcehre et al. 2016). They introduce a novel encoder to incorporate the conditioning information, additional loss functions to control the output, and perform a second round of training using policy gradient. A visualisation of the network architecture is given in Figure 3.1.

3.2.1 Task definition

The training data consists of context-question-answer triples $(\mathbf{D}, \mathbf{Q}, \mathbf{A})$, that have been tokenised such that $\mathbf{D} = \{d_1, d_2 \dots d_{|D|}\}$ where $|D|$ is the number of tokens in the document, and similarly for the question and answer.

[1]The order in which these operations are performed is important in practice, since the tokeniser uses case information to perform more accurate tokenisation.

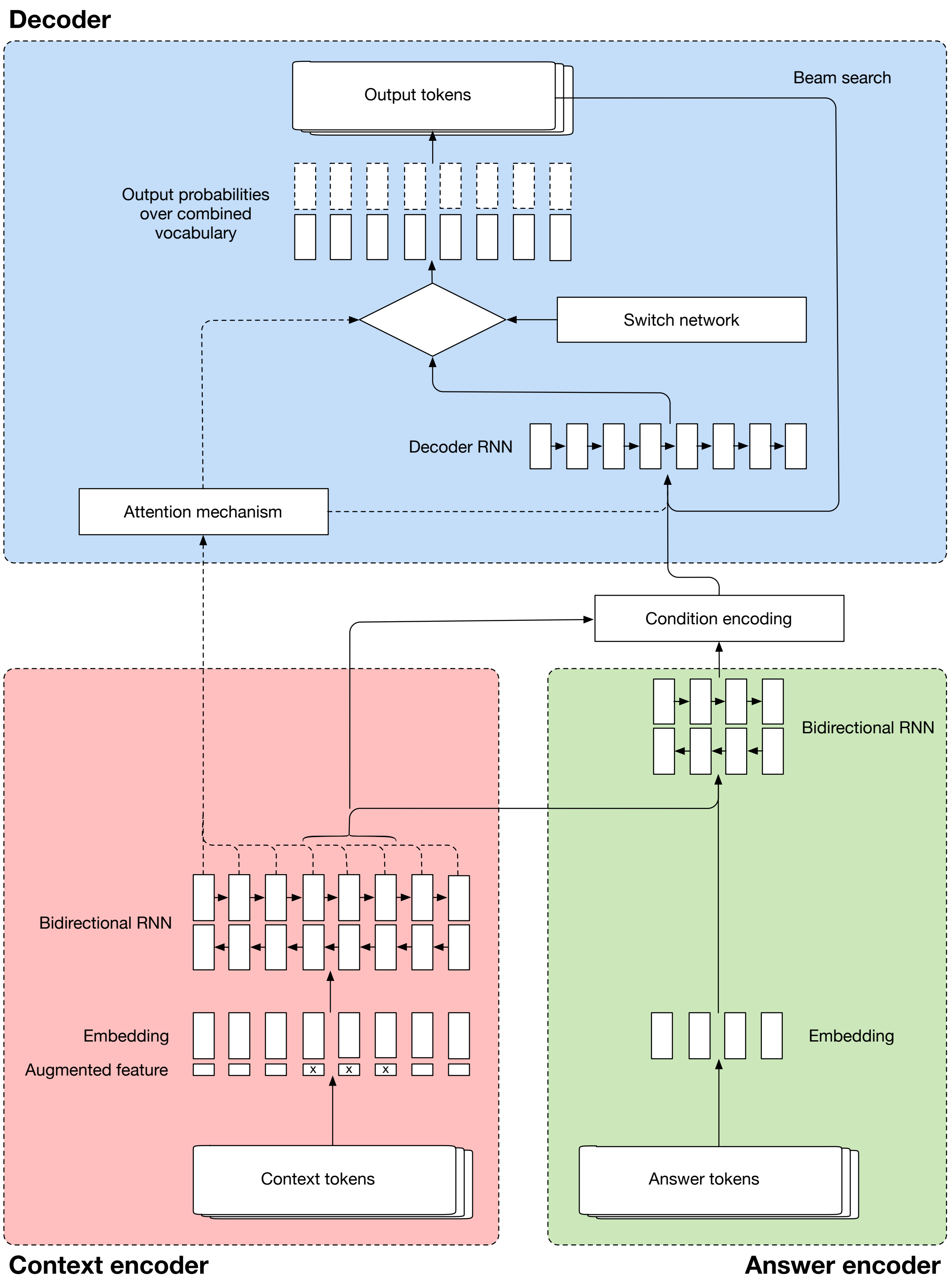


Figure 3.1: Model architecture diagram

The task is to generate a sequence of tokens $\hat{Y} = y_1, y_2 \dots y_T$, conditioned on a document $\mathbf{D}$ and answer $\mathbf{A}$ as input, by sampling from the parameterised conditional distribution at each time step given by

$$p(y_t) = p_\theta(y_t | y_{<t}, \mathbf{D}, \mathbf{A}). \tag{3.1}$$

3.2.2 Encoder

The location of the answer within the context is provided in SQuAD as the index of the first character of the answer, and along with the answer text this is used to calculate the indices of the context tokens that form the answer. During training, the ground truth question is also tokenised.

The context tokens are then transformed into an embedding representation $\mathbf{d}_t$, by looking up the relevant entry in the word embedding matrix. This matrix is defined for a shortlist vocabulary of 2000 tokens, plus special tokens[2]. We initialise with vectors from GloVe (Pennington et al. (2014)) if the word exists in the GloVe vocabulary, otherwise we initialise with a vector according to the Glorot initialisation scheme (Glorot and Y. Bengio 2010). This shortlist vocabulary is found by taking the top 2000 words in the SQuAD contexts and questions combined, ranked by frequency. Context words not in this shortlist are mapped to OOV for the purpose of embedding.

The embedded tokens $\mathbf{d}_t$ are augmented with an additional binary feature, indicating whether that position comprises part of the answer or not, so that $\tilde{\mathbf{d}}_t = [\mathbf{d}_t; \mathbb{I}(d_t \in \mathbf{A})]$.

The sequence of augmented embeddings is passed through a recurrent bidirectional neural network to generated the context encodings $\mathbf{h}^d_t$, whose state updates according to the LSTM function (Hochreiter and Schmidhuber 1997)

$$\mathbf{h}^d_t, \mathbf{s}_t = \text{LSTM}(\mathbf{s}_{t-1}, \mathbf{h}^d_{t-1}, \tilde{\mathbf{d}}_t). \tag{3.2}$$

The outputs of the two RNNs at each step are then concatenated.

The encodings at the time steps corresponding to the answer span are concatenated with

[2]Start-of-sentence, end-of-sentence, padding and out-of-vocabulary tokens are assigned their own embedding.

the embeddings of these tokens, and this (shorter) sequence is passed through a second bidirectional LSTM layer. The condition encoding vector $\mathbf{h}^a$ is then given by the concatenations of the output at the last time step of each RNN.

The initial state for the decoder RNN is calculated according to

$$\mathbf{r} = \mathbf{L}\mathbf{h}^a + \frac{1}{n}\sum_{t}^{|D|}\mathbf{h}_t^d, \tag{3.3}$$

$$\mathbf{s}_0 = \tanh(\mathbf{W}_0\mathbf{r} + \mathbf{b}_0), \tag{3.4}$$

where $\mathbf{L}$,$\mathbf{W}_0$, and $\mathbf{b}_0$ are learned parameters.

3.2.3 Decoder

The decoder is effectively a conditional language model, that outputs a distribution at each time step over possible output tokens from the combined shortlist and context vocabularies. It is a recurrent neural network, whose internal state updates according to the standard LSTM function such that

$$\mathbf{o}_t, \mathbf{s}_t = \text{LSTM}(\mathbf{s}_{t-1}, \mathbf{o}_{t-1}, \mathbf{v}_t). \tag{3.5}$$

At each time step, a weighted context vector $\mathbf{v}_t$ is computed using an attention mechanism to calculate soft alignments with the context document, and taking a weighted sum of the context encodings. This vector

$$\mathbf{v}_t = \sum_{t'}^{|D|}\alpha_{t,t'}\,\mathbf{h}_{t'}^d \tag{3.6}$$

is then used as the input to the decoder LSTM at each time step, to generate the outputs.

The alignment scores α_t are calculated with the standard Bahdanau attention mechanism (Bahdanau et al. 2014). Briefly, this takes the form of a fully connected network with a single hidden layer and a softmax output layer, that takes the current context encoding and previous hidden state as inputs, and produces a distribution over context time steps

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_j \exp(e_{t,j})} \tag{3.7}$$

as output, where

$$\mathbf{e}_t = f(v_t, s_{t-1}) \tag{3.8}$$

is a fully connected neural network with $\tanh$ activation.

To calculate the distribution over shortlist tokens $\hat{\mathbf{y}}_t^s$, the output of the LSTM cell at each step is projected into the dimensionality of the shortlist vocab, and normalised with a softmax activation, so that

$$\hat{\mathbf{y}}_t^s = \text{softmax}(\mathbf{W}_o \mathbf{o}_t + b_o). \tag{3.9}$$

The distribution over context locations for the pointer network $\hat{\mathbf{y}}_t^c$ is calculated by reusing the alignments from the attention mechanism α_t.

The combined distribution is then calculated by concatenating the shortlist and pointer outputs, weighted by a switch variable that controls the amount of mixing of the two distributions. This switch variable z_t is calculated at each step by passing $\mathbf{s}_t$, $\mathbf{v}_t$ and $\mathbf{y}_{t-1}$ as inputs to a feedforward network with two hidden layers and $\tanh$ activation, with a single output variable passed through a sigmoid activation, $z_t = \sigma(f(\mathbf{s}_t, \mathbf{v}_t, \mathbf{y}_{t-1}))$.

The final output distribution is therefore given by

$$\mathbf{p}_t = [z_t \hat{\mathbf{y}}_t^s;\ (1 - z_t)\alpha_t]. \tag{3.10}$$

The switch variable z_t behaves much like a latent variable in a mixture model. For datasets where the shortlist and copy vocabularies do not overlap, this latent variable effectively becomes observed, since we know at training time whether a word should be generated or copied. In practice this is rarely the case, and the vocabularies may overlap significantly (and indeed there may be repetition within the context) - we investigate this behaviour and its implications in detail in Chapter 4.

For training, we used the teacher forcing technique, whereby the ground truth token is used as input in place of the previously generated token. This prevents errors from compounding by optimising one-step-ahead predictions.

For evaluation, we used beam search (Graves 2012) to find sequences that have a high likelihood as a whole. We also zeroed out the probability mass for the out-of-vocabulary token, to force the decoder to generate valid words.

We trained the model on a maximum likelihood objective via minibatch gradient descent, using the Adam (Kingma and J. Ba 2014) optimisation algorithm. After each epoch of training, we evaluated the negative log likelihood (NLL) of the development set questions, and stopped training once the NLL reached a minimum[3].

Following Yuan et al. (2017), we included additional losses to encourage diversity in the output and to discourage the inclusion of words from the answer span. We performed ablation studies to confirm that these additional losses are beneficial.

The diversity loss takes the form of the entropy of the output distribution, given in Equation 3.11.

$$\mathcal{L}_e = \lambda_e \sum_t \mathbf{p}_t^T \log \mathbf{p}_t \tag{3.11}$$

To discourage the model from including words from the answer span in the output, we used the soft suppression constraint given in Equation 3.12.

$$\mathcal{L}_s = \lambda_s \sum_t \sum_{a \in \mathcal{A}} p_\theta(y_t = a | y_{<t}, D, A), \tag{3.12}$$

where $\mathcal{A}$ is the set of tokens forming the answer span.

[3]This hindsight is achieved by keeping copies of the parameters at each evaluation point, and discarding all but the version that gives the optimal NLL on the development set.

3.3 Implementation

No implementation of the model was released by Yuan et al. (2017), and so we implement the model from scratch, making extensive use of the functionality provided by the TensorFlow (Martin Abadi et al. 2015) library. This implementation is publicly available[4].

While TensorFlow provides built-in components to implement Bahdanau attention mechanisms and beam search decoding, it does not provide a pointer network implementation. We implemented the pointer network as a custom output layer for a RNN decoder, allowing us to continue to use TensorFlow's inbuilt decoders. This *copy layer* takes the output of the RNN and hidden state as input, and performs all the calculations described above to generate the output distribution over the combined vocabularies. This component is also reuseable, and may be used for future work on other tasks.

We note that it is important when dealing with minibatches of variable length sequences to correctly implement the mean operation, and to select the correct index when retrieving the last element in a sequence. The minibatch will be padded to accomodate the longest sequence in the batch, and so naively reducing a tensor along an axis will include these padding values. Instead, we made sure to implement the mean operation by taking the sum over the required axis, and normalising by the known length of each sequence. Likewise, when retrieving the last output of an RNN, we make sure to select the index corresponding to the final valid element in the sequence.

The hyperparameter settings are given in Table A.1. We trained the models for between 40,000 and 60,000 steps, depending on the model configuration.

Neural networks are notoriously hard to debug and test - their nature means that the model will often learn how to circumvent a small error, producing valid output but with a drop in performance. In order to check the behaviour of the model and gain some insight into its strength and weaknesses, we periodically rendered a visualisation of the output of the model, showing the value of the switch variable and the predicted and ground truth tokens at each time step. As shown in Figure 3.2, this allowed us to confirm that the model was indeed learning to copy when the training data indicated that it should. It also shows that the model is more

[4]`https://github.com/bloomsburyai/question-generation`

certain about generating from the shortlist at the start of the question (when the word is highly likely to be a “wh" word, irrespective of the inputs) compared to the middle of the question.

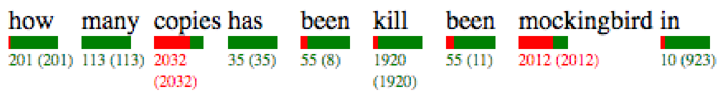


Figure 3.2: Example output of debug script part way through training. The bar indicates the value of the switch variable, with the predicted token ID shown alongside the ground truth token ID (in brackets).

Other checks included verifying that outputs that should be distributions were indeed normalised (e.g. the switch variable) and logging all auxiliary losses.

3.4 Interactive demo

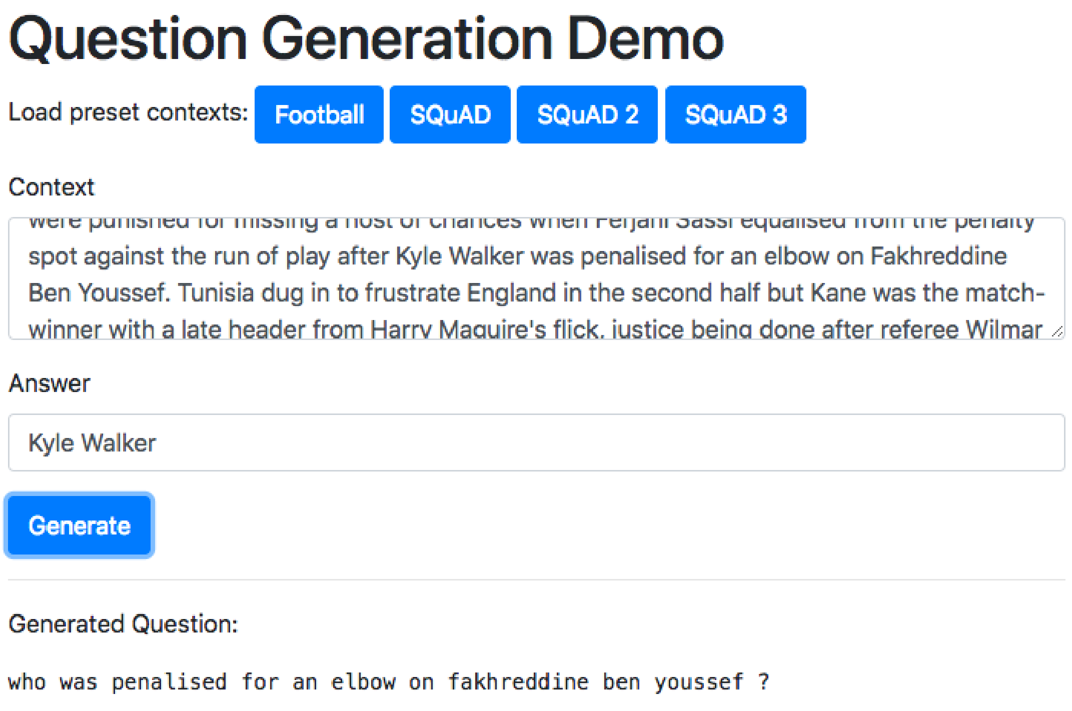
In order to test the behaviour of the model on inputs drawn from different domains and writing styles, we built a demo app that allows a user to input a context document as free text, select an answer within that context, and generate a question based on these inputs. This also allowed us to examine the behaviour of the models when presented with junk input, for example using “the" as the answer. A screenshot of this app is show in Figure 3.3 and a live instance may be found at `http://qgen.tomhosking.co.uk/`.

Briefly, the app is built using the Flask framework, and loads a global instance of the model that is loaded into memory once. Queries are then served synchronously. Since the model has already been loaded into memory, each query requires only preprocessing and a single forward pass through the model (including beam search), and so response latency is relatively low. Deployment in a production environment would require further work to avoid the synchronous bottleneck.

3.5 Experiments

3.5.1 Context cropping

Early experiments using the demo app showed that, for long documents with repeated words and answers without an obvious question, the model seemed to degenerate into a language model conditioned on the words that most frequently appeared in the context. An example



were punished for missing a host of chances when Ferjani Sassi equalised from the penalty spot against the run of play after Kyle Walker was penalised for an elbow on Fakhreddine Ben Youssef. Tunisia dug in to frustrate England in the second half but Kane was the match-winner with a late header from Harry Maguire's flick, justice being done after referee Wilmar

Kyle Walker

```
who was penalised for an elbow on fakhreddine ben youssef ?
```

Figure 3.3: Screenshot of demo application, allowing the user to generate questions interactively using custom context documents.

is given in Table 3.1, where in the absence of a good question to generate, the model has simply generated language by repeatedly using the word “england" that appears frequently in the context.

While there is no correct question that should have been generated here, the repetition of the word “english" implies that the encoder is in some way trying to represent the *whole* context document, not just the context local to the answer. We observe that although the context documents are generally formed of a number of sentences, the questions usually only refer to information within one of these.

We modify the model to first preprocess contexts by cropping them around the answer span. Initially we experimented with removing all sentences except the one containing the answer, but found that a few of the contexts in SQuAD are essentially lists of events separated by commas, giving a maximum sentence length of over 400 tokens.

We therefore additionally crop the contexts to a maximum of 200 tokens around the answer span. From inspecting the data, we found that some training questions used information from

Context
harry kane 's stoppage-time winner ensured england started their world cup campaign with victory after tunisia threatened to snatch a point in volgograd . kane scored his second goal of the game with a clever header as gareth southgate 's side recorded england 's first win in the opening game of a major tournament since they beat paraguay in the 2006 world cup . england 's captain gave them the reward they deserved for a brilliant start by turning in the opener in the 11th minute after tunisia keeper mouez hassen , who went off injured in the first half , clawed out john stones ' header . england ran tunisia ragged in that spell but were punished for missing a host of chances when ferjani sassi equalised from the penalty spot against the run of play after kyle walker was penalised for an elbow on fakhreddine ben youssef . tunisia dug in to frustrate england in the second half but kane was the match-winner with a late header from harry maguire 's flick , justice being done after referee wilmar roldan and the video assistant referee (var) had failed to spot him being wrestled to the ground twice in the penalty area . england play panama , who lost 3-0 to belgium earlier on monday , in their next group g game on sunday , which kicks off at 13:00 bst and will be shown live on the bbc .
Answer
by
Generated Question
how did england and england and england ?

Table 3.1: Example of a “junk" input answer, revealing the behaviour of the model under uncertainty.

multiple adjacent sentences, and so we also experiment with including one sentence from either side of the sentence containing the answer span.

This design decision essentially represents our prior belief that the question should be generated using information from the local vicinity of the answer span - by reducing the degrees of freedom of the model, we would expect the model to learn the task more easily.

An additional benefit of cropping the contexts is the reduced computational load. The maximum context length in SQuAD, without cropping, is 768 tokens; this is reduced to just over 200 tokens with cropping. The reduced memory footprint allowed us to increase the batch size during training, which in turn improved convergence speed. Combining these factors, cropping led to approximately a 4x increase in speed during training.

3.5.2 Ablation study

In order to investigate the importance of the combined shortlist and pointer vocabularies, we also performed experiments where we force the switch network either to use only the shortlist or only the pointer vocabularies.

We also trained models using simpler condition encoding, to evaluate the importance of the more advanced condition encoding. In this version of the model, the context encodings at the time steps corresponding to the answer span are averaged, and this vector used as the condition encoding vector, so that

$$\mathbf{h}_a = \frac{1}{|A|} \sum_{t \in \mathbf{A}} \mathbf{h}_t^d. \tag{3.13}$$

3.6 Evaluation

As discussed in Section 2.5.1, we make use of a number of automatic metrics. We briefly describe what they mean in the context of the AQ task. All scores are evaluated on our test split.

3.6.1 Metrics

3.6.1.1 F1

The mean F1 score between ground truth and generated questions measures the level of overlap in tokens, and is invariant to word order.

3.6.1.2 NLL

The NLL refers to the *negative log likelihood* of the ground truth sequence under the model. A lower value indicates that the model fits the data better.

3.6.1.3 BLEU

The BLEU score measures the overlap of n-grams between ground truth and generated questions. We calculate the corpus-level BLEU.

3.6.1.4 SOWE

SOWE refers to the cosine similarity between the sum of word embeddings for the ground truth and generated questions, and attempts to measure the similarity in *meaning* rather than tokens between the two sequences.

3.6.1.5 QA

QA refers to the mean F1 score attained by a question answering model when it attempts to answer the generated questions, and measures the relevance of generated questions. It can also be seen as a form of reconstruction score, since we should be able to recover the answer that was originally used to generate a question.

3.6.1.6 LM

We measure the fluency of language by calculating the perplexity of generated questions under a pre-trained LSTM language model, and report this as the LM score. A lower perplexity indicates a more probable sequence.

3.6.1.7 Disc

We experiment with using a discriminator, described in detail in Section 5.3, to estimate the probability that a given sequence was generated rather than coming from the ground truth data. The discriminator score therefore measures the similarity of generated questions to the ground truth data, with a higher score indicating that a model was more successful at *fooling* the discriminator.

3.6.2 Results

In Table 3.2 we show the values of the automatic evaluation metrics for these models, and compare to other published results in Table 3.3. Some examples of generated questions are given in Tables 3.4, 3.5 and 3.6.

We compare models with and without the advanced condition encoding (ACE), and with the context cropped to varying numbers of sentences around the answer span[5]. We also perform some ablation studies by disabling various components of the model.

[5]The number refers to the number of additional sentences included either side of the sentence containing the answer. For example, 0 means we keep only the sentence containing the answer, while 1 means we also include 1 sentence either side for a total of 3 sentences.

Features						**Metrics**						
ACE	*Crop window*	*Shortlist enabled*	*Copy enabled*	*Entropy loss*	*Suppression loss*	*F1*	*NLL*	*BLEU*	*SOWE*	*QA*	*LM*	*Disc*
Baseline						20.1	-	9.4	85.3	27.8	1328	20.7
-	-	✓	✓	✓	✓	46.7	**37.6**	14.8	91.1	64.9	73.2	7.7
✓	-	✓	✓	✓	✓	**47.6**	37.9	**15.0**	**91.3**	65.9	69.6	9.5
-	1	✓	✓	✓	✓	45.9	40.6	13.2	90.6	64.5	65.5	7.1
✓	1	✓	✓	✓	✓	47.0	40.9	14.6	91.1	**67.5**	67.0	9.0
-	3	✓	✓	✓	✓	46.2	38.3	13.6	90.9	65.4	60.7	7.1
✓	3	✓	✓	✓	✓	47.1	39.0	14.2	91.0	66.5	62.8	8.3
-	5	✓	✓	✓	✓	46.0	38.1	13.7	90.9	64.9	59.1	7.2
✓	5	✓	✓	✓	✓	47.3	37.9	14.5	91.3	66.3	63.2	9.4
✓	-	✓	-	✓	✓	29.8	49.7	3.8	88.3	47.9	1299	33.1
✓	-	-	✓	✓	✓	4.5	96.4	-	-	20.9	6062	**99.1**
✓	3	✓	✓	-	✓	46.7	39.0	13.6	90.8	65.5	62.7	8.7
✓	3	✓	✓	✓	-	46.5	38.6	13.9	91.0	64.7	**58.1**	8.2
Ground truth						-	-	-	-	71.2	101.5	-

Table 3.2: Automatic evaluation metrics evaluated on various model configurations. The bottom row shows the scores attained by our QA and language models on ground truth questions. The best scores for each metric are highlighted in bold.

The baseline model refers to the PCFG-Trans rule-based system (Heilman 2011) as modified for SQuAD by Zhou et al. (2018).

The ACE improves many of the automatic metrics, although by only a slim margin of 0.2 BLEU points. Cropping the inputs to only the sentence containing the answer span improves the relevance of generated questions, with a QA system more likely to find the correct answer. Note that the full, un-cropped context was used as input for the QA system, and the improved score is not because the QA task was easier for the cropped models.

There is a reduction in performance for metrics other than QA when the context is cropped, although this loss is smaller for the models with ACE; cropping decreases the BLEU by 0.4 points for the model with ACE, compare to 1.6 points without. We conclude that the ACE allows the model to extract a more useful condition encoding, which is particularly important when there is less information available to the model.

The model without a pointer network manages to generate questions, but the small size of the shortlist vocabulary means that these are often low relevance, attaining poor scores in all metrics.

The model that only uses a pointer network struggles to generate meaningful sequences at all, and receives very poor scores in almost all metrics. The discriminator score appears anomalous, and we discuss it in more detail in Section 5.5.

Disabling the auxiliary entropy or suppression losses leads to a small reduction in performance of 0.6 and 0.3 BLEU points respectively.

The language model perplexity is generally better for generated questions than for ground truth questions. This is actually to be expected, as the generated questions are effectively samples from a conditional language model trained on the same data as the language model used to evaluate them. This is somewhat problematic, and we would ideally evaluate the language quality using a different approach. We discuss this further in Section 5.3.

Model	Source	BLEU
PCFG-Trans	Zhou et al. (2018)	9.31
Baseline	Our result	9.4
Seq2Seq + attn (no pointer)	Zhou et al. (2018)	3.06
Seq2Seq + attn (no pointer)	Our result	3.8
Advanced Condition Encoding (ACE)	Yuan et al. (2017)	10.2
Advanced Condition Encoding (ACE)	Our result	15.0

Table 3.3: Comparison of BLEU scores against other published results

The model with advanced condition encoding corresponds to the model labelled “Our model" in Yuan et al. (2017). They report a BLEU score of 10.2, while our implementation gives a BLEU of 15.0. We confirmed with the authors that we had replicated their intended model correctly, and believe the discrepancy may be due to the use of instance level BLEU scores rather than corpus level (see Section 2.5.1). Our evaluations of other configurations agree with the results given by Zhou et al. (2018), as shown in Table 3.3, supporting this theory.

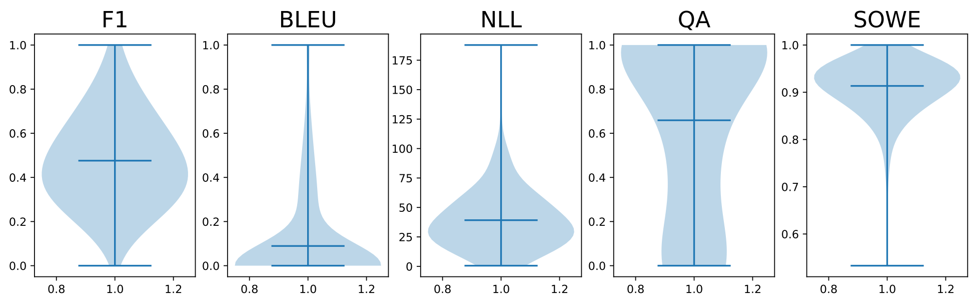


Figure 3.4: Distribution of scores for the uncropped model with ACE.

Figure 3.4 shows the distribution of various metrics across the test set. The instance-level BLEU score is heavily skewed towards zero, as discussed in Section 2.5.1.

Table 3.4 shows generated questions for a relatively easy example. Although none of the models is able to generate the ground truth question, most of them are able to generate a question that is essentially a paraphrase of the correct question. For this example, the model without pointer network is able to generate a very convincing question, despite only being able to use a shortlist vocabulary of 2000 words. The model that is *only* able to copy fails to generate a structured sequence at all.

Table 3.5 shows a much harder example - correctly generating the ground truth question requires coreference resolution to determine that “the two" refers to “l" and “p", which is itself a non-trivial task. Many of the models are nonetheless able to generate convincing questions. In this example, the subject matter is sufficiently different to that of the training examples that the model without copy mechanism is not able to understand the context, and fails to generate a structured sequence.

Table 3.6 reveals a curious effect - some of the models have generated questions about “Darwin" or “Gaddafi", despite there being no reference to these people in the context. These models have correctly attempted to ask a question about a person, but have failed to select the correct one. Initially, it is not obvious why they are able to generate these names at all - in fact, they form part of the shortlist vocabulary. We investigate this further in Chapter 4.

Context
from 2005 to 2014 , there were two major league soccer teams in los angeles — the la galaxy and chivas usa — that both played at the **stubhub center** and were local rivals . however , chivas were suspended following the 2014 mls season , with a second mls team scheduled to return in 2018 .
Answer
stubhub center
Ground Truth Question
what was the name of the stadium that the teams played in ?
Seq2Seq +attn +pointer
where were the la galaxy and chivas usa located ?
ACE
where did the la galaxy and chivas usa occur ?
Seq2Seq +attn +pointer, crop window = 1
where did the la galaxy and chivas usa play ?
ACE, crop window = 1
where did the la galaxy and chivas usa play ?
Seq2Seq +attn +pointer, crop window = 3
where were the la galaxy and chivas usa ?
ACE, crop window = 3
where did the la galaxy and chivas usa play ?
Seq2Seq +attn +pointer, crop window = 5
where did the la galaxy and chivas usa play ?
ACE, crop window = 5
where did the la galaxy and chivas usa ?
ACE, no pointer
where did the team play ?
ACE, no shortlist
usa —

Table 3.4: Example generated questions, showing similar results for an easy case.

Context
similarly , it is not known if l (the set of all problems that can be solved in logarithmic space) is strictly contained in p or equal to p. again , there are many complexity classes between the two , such as **nl and nc** , and it is not known if they are distinct or equal classes .
Answer
nl and nc
Ground Truth Question
what are two complexity classes between l and p ?
Seq2Seq +attn +pointer
what are the complexity classes between the two ?
ACE
what are the two classes classes between the two ?
Seq2Seq +attn +pointer, crop window = 1
what are two complexity classes between the two ?
ACE, crop window = 1
what are two complexity classes between the two ?
Seq2Seq +attn +pointer, crop window = 3
what type of classes are two complexity classes between the two ?
ACE, crop window = 3
what are two complexity classes between the two ?
Seq2Seq +attn +pointer, crop window = 5
what are the two complexity classes between ?
ACE, crop window = 5
what are the two complexity classes between the two ?
ACE, no pointer
what are the two classes ?
ACE, no shortlist
l complexity contained contained contained [...]

Table 3.5: Example generated questions for a less easy example.

Context
in december 1878 , tesla left graz and severed all relations with his family to hide the fact that he dropped out of school . **his friends thought that he had drowned in the mur river** . tesla went to maribor (now in slovenia) , where he worked as a draftsman for 60 florins a month . he spent his spare time playing cards with local men on the streets . in march 1879 , milutin tesla went to maribor to beg his son to return home , but nikola refused . nikola suffered a nervous breakdown at around the same time .
Answer
his friends thought that he had drowned in the mur river .
Ground Truth Question
what was tesla 's friends ' theory as to what became of him ?
Seq2Seq +attn +pointer
what did tesla ?
ACE
why did tesla when he had ?
Seq2Seq +attn +pointer, crop window = 1
what did darwin write about ?
ACE, crop window = 1
what did gaddafi 's friends think ?
Seq2Seq +attn +pointer, crop window = 3
what did tesla do ?
ACE, crop window = 3
why did tesla die ?
Seq2Seq +attn +pointer, crop window = 5
what did tesla do ?
ACE, crop window = 5
why did tesla die ?

Table 3.6: Example generated questions, showing strange shortlist vocabulary.

Chapter 4

Copy mechanism

The pointer network, and its ability to copy words from the context document, are crucial for the models ability to generate questions and to generalise to new domains. As shown in Table 3.2, although the pointer network alone is not sufficient to generate good questions, its inclusion in the model results in a large performance increase. The examples in Table 3.6 show that the details of the implementation are important.

We investigate the pointer network in detail, and perform a number of experiments and ablations to understand the implications of different implementation choices. While a number of different approaches have been proposed (Gu et al. 2016a; Gulcehre et al. 2016; See et al. 2017), we do not know of any existing work that has compared these approaches.

4.1 Shortlist vocabulary selection

Yuan et al. (2017) use the top 2000 words by frequency from the SQuAD training contexts and questions combined for the shortlist vocabulary. There are only 442 distinct context documents in the training data, and so this vocabulary is not a good proxy for the most frequent words in English in general. For example, “beyoncé" is the 445th most frequent word in SQuAD and therefore is included in the shortlist, but seems unlikely to be so highly ranked across all written English.

The choice of shortlist can potentially severely limit the ability of the model to generate good questions. Early experiments accidentally used a shortlist constructed using only the training contexts, which did not include a question mark - a major limitation for a question generation system.

Figure 4.1 shows the distribution of frequencies of the top 2000 words in the SQuAD questions. It appears to obey Zipf's Law, which states that the frequency of a word in a corpus is inversely proportional to its rank in the frequency table. Only a small set of words appear frequently in the training questions.

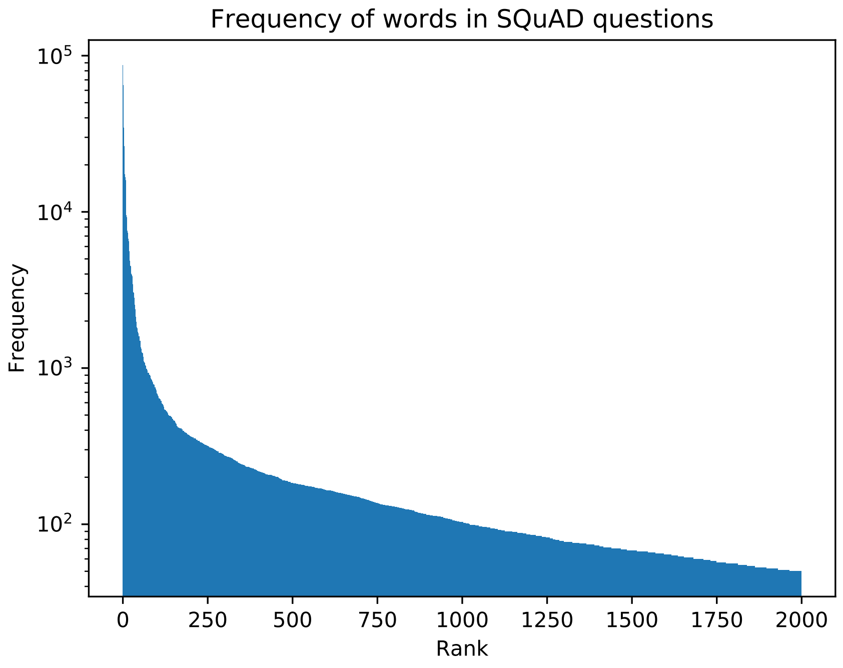


Figure 4.1: Plot of word frequency in SQuAD against the rank of that word in the frequency table.

4.1.1 GloVe

We experiment with an alternative choice of shortlist vocabulary, the top 2000 words from the GloVe embeddings (Pennington et al. 2014). These are collected from a total of 6 billion tokens (compared to the 13 million tokens that form the SQuAD training set), from the *Wikipedia 2014 + Gigaword 5* corpora, and are therefore a better representation of word frequency in English in general.

4.2 Implicit biases during training

The process of training the model requires that we specify a ground truth question, with each token in the sequence encoded as a one-hot vector in the combined shortlist and location vocabularies. For the models in Chapter 3, the training data was encoded according to very straightforward heuristic, in keeping with the original formulation by Gulcehre et al. (2016). If the token to be encoded exists in the shortlist vocabulary, it is encoded using that ID. If it does not, but it does exist in the context vocabulary, then the heuristic simply finds the first occurrence of the word in the context and encodes the token using that location.

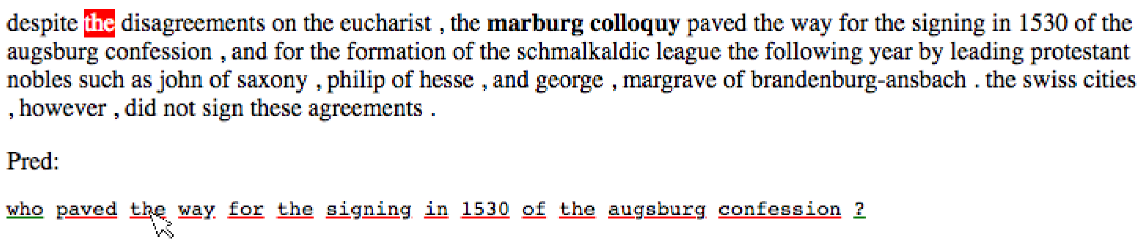
For examples where the context vocabulary is orthogonal to the shortlist vocabulary (i.e. there are no overlapping words) and the words in the context are not repeated, there is only one way to encode the question, and this heuristic is sufficient. The pointer network was originally used in the context of NMT, where the source and target languages are different and have vocabularies that are essentially non-overlapping. For the AQ task, the source and target languages are both English, and there is likely to be significant overlap between the shortlist and location vocabularies.

The task can to an extent be viewed as a hybrid between an extractive and abstractive task. At the extremes, we could either generate a question by only rearranging words from the context (i.e. only using the pointer network) or by generating a dense representation of the condition and only using that to generate a question (i.e. ignoring the attention and copy mechanisms). Intuitively, the former seems like an easier task for a model to learn, and is the approach taken by the non-neural models. We therefore experiment with switching the order of the heuristic, such that the location vocabulary is prioritised over the shortlist vocabulary, to bias the model to prefer copying over generating from the shortlist.

While Yuan et al. (2017) state that they use the same parameters for both the attention mechanism and pointer network, they do not explain this decision. There is some overlap between the tasks they are designed to achieve; the attention mechanism was originally designed to find alignments between source and target sequences (Bahdanau et al. 2014), but has also been shown to attend more generally to parts of an input when performing a task (Selvaraju et al. 2017). An attention mechanism can therefore allow the decoder to consider information from different parts of the context, a different task to identifying locations that should be copied.

We therefore experiment with a model that uses a separate attention mechanism and pointer network.

4.3 Copy location


despite the disagreements on the eucharist , the **marburg colloquy** paved the way for the signing in 1530 of the augsburg confession , and for the formation of the schmalkaldic league the following year by leading protestant nobles such as john of saxony , philip of hesse , and george , margrave of brandenburg-ansbach . the swiss cities , however , did not sign these agreements .

Pred:

`who paved the way for the signing in 1530 of the augsburg confession ?`

Figure 4.2: Screenshot of our debug page, used for inspecting the pointer network behaviour during inference. Words underlined in red were copied, while those underline in green were generated from the shortlist vocab. The answer is highlighted in bold. The mouseover shows that the token “the"" was copied from early in the paragraph, rather than from near the answer.

Using the first occurence of a word in the context as the location to copy from is trivial to implement, but seems unlikely to be optimal. Intuitively, we would expect the best source of information for generating the question to be centered around the answer span, as confirmed by the experiments in Chapter 3.

Although some abstractive behaviour might be desirable, this adds unnecessary complexity to the task. The context is already made up of well formed language, and so a model that performs a minimum of modification to this language might be expected to learn more easily than one that must learn to generate language from scratch. It may be better for the model to learn to copy contiguous phrases from the context if possible, only modifying where necessary.

Figure 4.2 shows the output of a debug tool for a model with copy priority, showing a generated question and the vocabulary used to generated each token. Hovering over each copied token shows the location from which it was copied. This example confirms that the model has learned to copy words from early in the text, even though the word “the" appears just after the answer span as part of a phrase that the model itself generated. This also shows that the model is not copying contiguous phrases from the context, and has instead attempted to learn the more difficult task of constructing these phrases from scratch.

We propose a modified “smart" heuristic for encoding word locations, given in Algorithm 1.

This heuristic is designed to encourage the model to use tokens near the answer, and to use contiguous phrases from the context where possible.

Algorithm 1 Smart copy heuristic

```
if the token exists only once in the context then
    use that location
else
    if there are multiple instances
    and the previously encoded token was from the location vocabulary
    and the token at the next position in the context is the correct token then
        use that position
    else
        find the occurence of the current token nearest the answer span
```

4.4 Set-based copy vocabulary

The methods stated so far effectively assume that there is only one correct location to copy from, and that all words in the context are unique. This assumption often does not hold, and there may exist multiple locations in the context with the same word. The network may therefore end up splitting probability mass between the correct word at different locations, resulting in the wrong word being copied.

We convert the list of context words that form the context into a set of unique words, to form a vocabulary that is non overlapping. The network therefore only has to find the correct word in the context, irrespective of its location.

4.5 Latent training

The strength of deep models is that they are able to learn feature representations directly from data, and so it makes sense that we should aim to remove as much bias as possible from the training data. As we have already commented, if we do not take into account repeated words in the context, the model may be forced to split probability mass between different occurrences of the same word, resulting in poor predictions. This is also true for the case where words appear both in the context and the shortlist vocabulary.

The switch variable can be viewed as a hidden latent variable whose correct value at each time step is unknown, where only the final observed prediction should be used to calculate the loss. We therefore train a model where the training data does not bias the switch either way, and the value of the switch variable is learned implicitly.

We achieve this by using encoding each token as a k-hot vector, where any index that corresponds to the correct word is set to 1, across both the shortlist and location vocabularies. The cross entropy loss is then calculated by summing the probabilities for all valid outputs, before taking the log as normal. This loss therefore does not distinguish between the different ways that the model can produce the correct word, and we refer to this approach as latent training.

We also experiment with increasing the size of the hidden layers of the switch network for the latent case, to test whether this might be a bottleneck in making good decisions regarding when to copy.

4.6 Evaluation

Table 4.1 shows the automatic evaluation metrics for a number of variations on the implementation of the copy mechanism. We also repeat select results from Chapter 3 for convenience, along with the performance of LM and QA models on the ground truth data.

Somewhat counter-intuitively, the performance of the model with a smart copy heuristic is lower than with the naive heuristic. While the reasoning behind the smart approach seems sounds, the bias it introduces evidently hinders learning rather than improving it. Biasing the model to prioritise copying rather than the shortlist leads to a significant drop in performance, although the language score does improve slightly.

The model that uses a GloVe shortlist vocabulary shows slightly improved perplexity under the language model, but is otherwise marginally worse.

Training a model without biasing the switch variable leads to the highest F1 score, and also gives significant improvement in the QA score, with only a small drop in BLEU score. Table 4.2 show the mean probability of generating a token by copying rather than from the shortlist. As expected, the model trained with copy priority is most likely to copy. Removing the bias from the training data allows the model to learn the right balance, with the copy probability falling between the copy priority and normal (shortlist priority) models.

It can also be seen that including the smart copy heuristic lowers the probability of copying, implying that this heuristic prevents the model from learning to copy and thus hinders performance.

[1]This model used a larger network for the switch variable

Features							Metrics						
Crop window	*Smart copy*	*GloVe shortlist*	*Set vocab*	*Latent training*	*Copy priority*	*Separate copy/attn*	*F1*	*NLL*	*BLEU*	*SOWE*	*QA*	*LM*	*Disc*
Baseline							20.1	-	9.4	85.3	27.8	1328	20.7
-	-	-	-	-	-	-	47.6	37.9	15.0	91.3	65.9	69.6	9.5
3	-	-	-	-	-	-	47.1	39.0	14.2	91.0	66.5	62.8	8.3
-	-	✓	-	-	-	-	47.0	38.1	14.6	91.2	65.4	62.3	6.0
-	-	-	-	✓	-	-	**47.9**	39.4	14.8	91.2	68.1	66.7	7.2
-	-	✓	-	✓	-	-	47.1	38.7	14.6	91.3	64.7	61.7	6.1
3	-	-	-	-	-	✓	47.0	38.8	14.1	91.0	64.8	60.8	9.0
3	✓	-	-	-	-	-	46.8	38.8	14.0	91.1	66.4	**56.8**	7.5
3	✓	-	-	-	✓	-	44.7	43.3	10.6	90.1	59.8	57.7	5.9
3	-	-	✓	-	-	-	45.1	**37.9**	13.6	**92.4**	**71.0**	155	**24.3**
3	-	-	-	✓	-	-	47.0	39.2	13.9	91.0	66.3	58.8	8.2
3	-	-	-	✓^{1}	-	-	47.2	39.1	14.1	91.1	66.7	65.2	6.9
3	-	-	-	✓	-	✓	46.6	39.2	13.9	90.8	65.8	67.8	7.7
3	-	✓	-	✓	-	-	47.2	39.6	14.8	91.2	66.2	64.7	5.0
3	-	✓	✓	-	-	-	47.4	38.2	**16.3**	91.8	69.0	91.1	9.3
Ground truth							-	-	-	-	71.2	101.5	-

Table 4.1: Automatic evaluation metrics evaluated on various configurations of the copy mechanism. The bottom row shows the scores attained by our QA and language models on ground truth questions. The best scores for each metric are highlighted in bold.

Increasing the capacity of the model by making the switch network larger or separating out the copy and attention mechanism does not make a significant difference, implying that the model primarily attends to the same part of the context that it is copying.

The model that converts the context to a set of distinct words achieves high NLL, SOWE and an increase of 4.5% in the QA score. The examples in Table 4.3 give some insight into why - this model has learned to copy long sections of text from the context document. This makes the generated questions more specific to the region of the context around the answer, which in turn improves the ability of a QA model to find that answer. However, it also attains the

Model type	Mean copy probability
No crop	0.150
Cropped	0.148
+ smart heuristic	0.136
+ copy priority	0.465
Latent training	0.156

Table 4.2: Copy probabilities for different models

worst LM score, implying that the language it produces is considered as less probable by the language model. Qualitatively, this model seems to copy continuous phrases from the context more often, which may be different to the questions that were used to train the language model.

The model that uses both a set-based copy vocabulary and a GloVe shortlist vocabulary achieves the highest BLEU score, improving on the uncropped ACE model by 1.3 points, and very competitive QA, F1 and NLL scores.

In general, the differences in performance are relatively small, and the examples shown in Table 4.3 and 4.4 have only minor differences. It is only the model with copy priority that achieves a significantly worse score. We conclude that the pointer network is a relatively robust approach to generating language, that is not overly sensitive to the specific implementation. While we have shown that the scores can be improved by optimising this aspect of the model, many of the less optimal configurations are still able to produce good questions.

When the GloVe shortlist and latent training are both used for a model with cropping, they bring performance back up to almost the same level as an uncropped model. As discussed in Section 3.5.1, cropping improves training speed by a factor of 4, which is an important consideration for the policy gradient experiments discussed in Chapter 5. Qualitatively, the questions generated by this model appear to be slightly more fluent and specific than other models.

Due to time constraints, this configuration was the best available model when we selected a starting point for fine tuning with policy gradients, and we therefore use this configuration as our starting point for policy gradient training. The results from our fine tuning experiments should therefore be considered relative to the starting model, rather than in absolute terms.

Context
the **education service contracting** scheme of the government provides financial assistance for tuition and other school fees of students turned away from public high schools because of enrollment overflows . the tuition fee supplement is geared to students enrolled in priority courses in post–secondary and non–degree programmes , including vocational and technical courses . the private education student financial assistance is made available to underprivileged , but deserving high school graduates , who wish to pursue college/technical education in private colleges and universities .
Answer
education service contracting
Ground Truth Question
what is the name of the scheme that provides tuition and fee assistance to students due to excess enrollment ?
ACE
what contracting of the government provides financial assistance ?
ACE +latent training
who provides financial assistance for tuition ?
ACE +smart copy, crop window = 3
who provides financial assistance for tuition ?
ACE +copy priority +smart copy, , crop window = 3
what contracting scheme of the government ?
ACE +set vocab, crop window = 3
who provides financial assistance for tuition and other school fees of students turned away from public high schools instead of enrollment overflows ? ”
ACE +latent training, crop window = 3
who provides scheme of the government ?
ACE +latent training +GloVe shortlist, crop window = 3
what part of the government provides financial assistance for tuition ?
ACE +set vocab +GloVe shortlist, crop window = 3
what contracting scheme of the government provides financial assistance for tuition ?

Table 4.3: Example generated questions for various approaches to the copy mechanism.

Context
while primary chloroplasts have a **double** membrane from their cyanobacterial ancestor , secondary chloroplasts have additional membranes outside of the original two , as a result of the secondary endosymbiotic event , when a nonphotosynthetic eukaryote engulfed a chloroplast-containing alga but failed to digest it - much like the cyanobacterium at the beginning of this story . the engulfed alga was broken down , leaving only its chloroplast , and sometimes its cell membrane and nucleus , forming a chloroplast with three or four membranes - the two cyanobacterial membranes , sometimes the eaten alga 's cell membrane , and the phagosomal vacuole from the host 's cell membrane .
Answer
double
Ground Truth Question
what kind of membrane do primary chloroplasts have ?
ACE
secondary chloroplasts did primary chloroplasts have ?
ACE +latent training
what does secondary chloroplasts have ?
ACE +smart copy, crop window = 3
what do primary chloroplasts have ?
ACE +copy priority +smart copy, , crop window = 3
what kind of membrane was primary chloroplasts ?
ACE +set vocab, crop window = 3
what kind of membrane do secondary chloroplasts have from their cyanobacterial ancestor ? .
ACE +latent training, crop window = 3
what do primary chloroplasts have ?
ACE +latent training +GloVe shortlist, crop window = 3
what kind of membrane do primary chloroplasts have ?
ACE +set vocab +GloVe shortlist, crop window = 3
what do primary chloroplasts have from their cyanobacterial ancestor ?

Table 4.4: Example generated questions for various approaches to the copy mechanism.

Chapter 5

Non differentiable training objectives

As observed in Section 2.5.2, the problem of generating questions is a one-to-many problem, with multiple valid questions and phrasings for each input. Additionally, the process of teacher forcing leads to exposure bias, where the model is unable to learn from its mistakes. In the absence of multiple ground truth questions for each answer, it is therefore desirable to optimise the model directly on its performance, and to decouple it from the training data.

In this chapter, we follow Yuan et al. (2017) and fine tune a model to optimise rewards from a QA and language model. We also experiment with a novel discriminator that learns the reward directly from data, and with performing adversarial training where this discriminator is updated concurrently with the question generator.

5.1 QA and LM

Yuan et al. (2017) argue that the question generation task can essentially be broken down into two components: generating good language and generating a string that, when interpreted as a question, gives the same answer as was used to generate it. This is reminiscent of the fluency and adequacy criteria used to assess machine translation quality (e.g. (Bojar et al. 2016; Koehn and Monz 2006)). To this end, they propose tuning the model to maximise these qualities by using additional models as proxies, after an initial period of maximum likelihood training.

Specifically, the model is used to generate a complete question, which is evaluated and a reward calculated. The parameters of the model are then updated to maximise this reward. We assume for now that this reward is a good proxy for the question quality, and investigate this assumption in Chapter 6.

As discussed in Section 2.3.3, the use of beam search means we cannot calculate the gradient of the reward with respect to the parameters, i.e. rewards calculated on generated sequences are non-differentiable. REINFORCE (Williams 1992) can instead be used to train the model by treating question generation as a sequential decision process, and updating this decision policy.

To calculate the rewards, we use a pre-trained question answering system and a pre-trained language model.

5.1.1 Language model reward

We use a LSTM language model, composed of an embedding lookup layer, a unidirectional LSTM layer, and a dense output layer with softmax activation. Layer norm (J. L. Ba et al. 2016) and dropout (Srivastava et al. 2014) are applied to the recurrent layer. This model is trained via minibatch gradient descent to maximise the conditional log likelihood of each token in the sequence, given the previous tokens. The hyperparameters for the language model are given in Table A.2.

Given a generated question, we calculate the reward as the negative perplexity of this sequence under the language model, as given in Equation 2.6. We note that the language model assigns a lower perplexity to our test set split than to the deveopment set, implying that the ground truth questions in the development set use more complex language than those in the test split.

	PPL
Dev	114.44309
Test	101.47814

Table 5.1: Perplexities of the questions in our SQuAD development and test set splits, under the LSTM language model.

5.1.2 Question answerability reward

To assess the adequacy or relevance of a generated question, we feed it as input to an automatic question answering system. We then use the F1 score of the estimated answer compared to the ground truth answer span as the reward. This can be seen as a form of reconstruction loss; the question was generated conditioned on the answer span, and should be specific and relevant enough that it is possible to recover this answer.

Since the QA model will be used as a performance metric and a reward generator, QANet (A. W. Yu et al. 2018) appeals for two reasons. Firstly, its high performance means that the reward signal will be less noisy, as an objectively valid question is more likely to be answered correctly by the model. Secondly, as a convolutional model it is computationally efficient, making it less expensive to interleave into the training loop of another model. We use the open source implementation of QANet in TensorFlow published by Kim (2018). The full model architecture is shown in Figure 5.1.

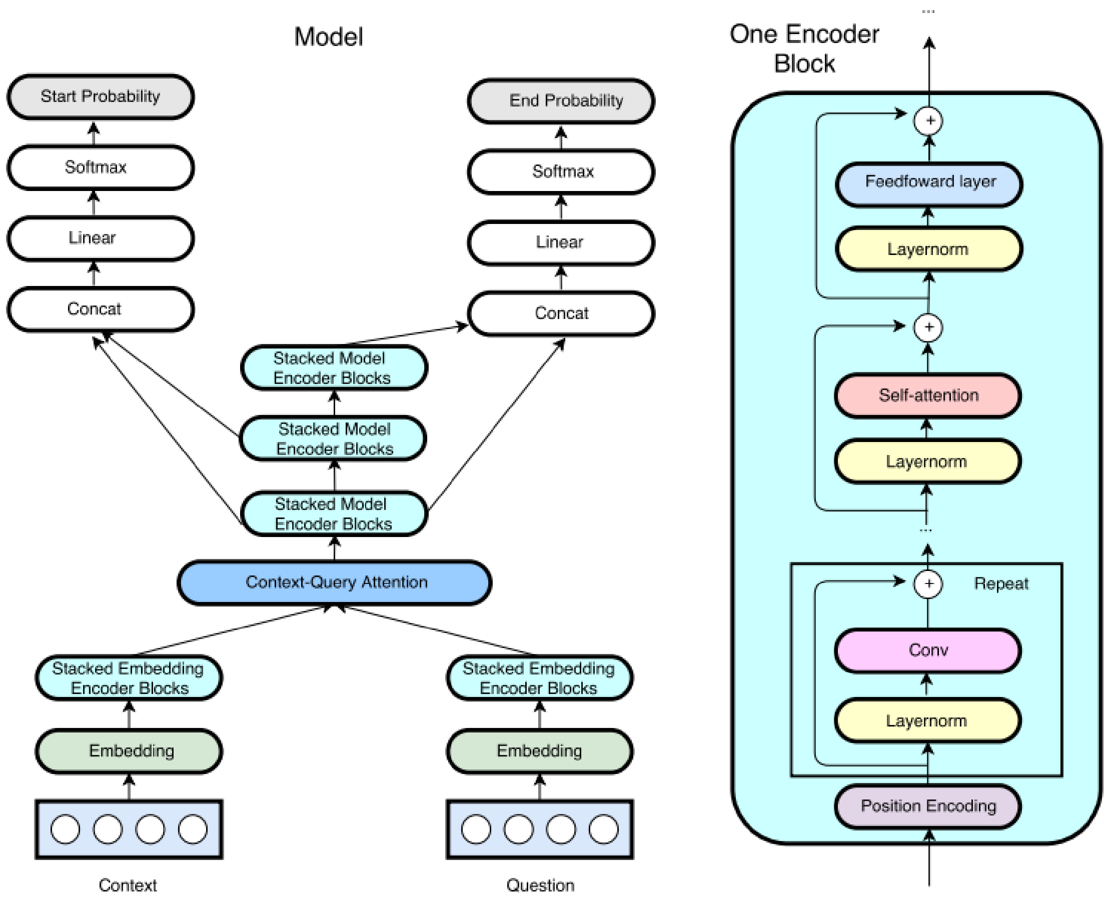


Figure 5.1: QANet architecture diagram (A. W. Yu et al. 2018)

In order to reduce memory use such that the model could be trained on a single 12GB GPU, we used the reduced size model proposed by Kim (2018), with a single attention head rather than 8, and with 96 hidden units instead of the original 128. The performance of the trained model on the ground truth questions in our SQuAD development and test set splits is shown in Table 5.2. We report the *exact match* score, which measures the percentage of answers that were correct, and the F1 score, which measures the word overlap between predicted and ground truth answers. Our scores are similar to those reported by Kim (2018), but are not comparable due to the different dev/test split used.

The scores are calculated according to the method used for the SQuAD leaderboard (Rajpurkar et al. 2016), which normalises answers for punctuation, articles and capitalisation before calculating the score[1]. Additionally, multiple valid answers are provided in the SQuAD development set, corresponding to the various answers given by the crowdworkers. The score for each context-question pair is given by the maximum over these possible answers. Since the question generator only uses a single context-answer pair as input, we also report the scores for the QA model using only the first possible answer, and use this as the baseline when evaluating the answerability of generated questions.

	Exact Match (EM)	F1 score
Dev	67.1	77.1
Test	65.5	76.8
Test (single answer)	55.6	71.2

Table 5.2: QANet results on our SQuAD development and test set splits.

5.1.3 Training

We fine tune models using either the QA or the LM reward, and also using both concurrently. Weightings for the rewards are given in Table A.4.

The process of beam searching does not maintain all the activations used to generate the final sequence. In order to replicate this set of activations so that backpropagation can occur, we use the generated sequence to teacher force the decoder. The REINFORCE loss given in Equation 5.1 can then be applied to the teacher forced outputs, and the parameters updated using the Adam optimiser. In other words, the teacher forcing is used to select the policy that

[1]Specifically, punctuation and the words “a", “an" and “the" are removed, and the string is converted to lowercase.

was actually chosen by beam search, and update the corresponding probability with the reward it achieved.

$$\nabla\mathcal{L} = \nabla\frac{1}{l}\sum_t\left(\frac{R(\hat{Y}) - \mu_R}{\sigma_R}\right)\log\pi(\hat{y}_t|\hat{y}_{<t}, D, A) \tag{5.1}$$

5.2 Stability

In order to improve stability of the policy gradient training and to prevent the model from forgetting what it has learned during the maximum likelihood phase, we follow Yuan et al. (2017) and continue to train using a maximum likelihood objective on the ground truth data. Each minibatch is then formed of an equal number of reward samples and maximum likelihood samples.

We experimented with removing this maximum likelihood objective and training solely using REINFORCE, and confirmed that training without this objective is indeed unstable, with the model rapidly degenerating and producing unstructured sequences.

It is worth noting that the REINFORCE loss has a very similar form to the maximum likelihood cross-entropy loss. Implementing this concurrent training is therefore quite straightforward, as the cross-entropy loss can be recovered by feeding the ground truth question as input, with a reward of 1. Instead of selecting the policy that *was* used and updating according to the reward, cross-entropy effectively updates the policy that *should* have been used, with unit reward.

The μ_R and σ_R in Equation 5.1 refer to the mean and standard deviation of the rewards so far, which can be calculated online. Yuan et al. (2017) found that some form of whitening was important for training stability, and use a simple form of PopArt Hasselt et al. (2016) to de-mean and normalise the rewards. Initial experiments using our model confirmed this, and we found that using un-normalised rewards led to catastrophic forgetting, with the model rapidly failing to generate any structured sequences.

Additionally, we found that we had to introduce two *burn in* periods to the second (policy gradient) phase of training.

Firstly, we implemented a simple learning rate schedule, which increased the learning rate linearly from 0 to the normal value of 2×10^{-4} over the course of the first 200 steps. Without this, the initial few steps were based on a noisy signal from the REINFORCE loss, and were not smoothed by the Adam optimiser as its internal momentum calculating had not yet had a chance to calibrate to the scale of the gradients.

Secondly, we found it necessary to let the rolling mean and variance calculations reach a point of relative stability, by generating samples for a number of steps[2] and updating these rolling statistics before applying any parameter updates to the model.

In order to prevent the generator from cheating and learning to encode the answer directly in the question, we implemented a hard form of answer suppression loss, by masking out the answer span in the location softmax for questions that were passed to the QA model.

5.3 Discriminator

Although the use of a language model and QA system as reward generators makes intuitive sense, and reflects existing evaluation methods for NMT (Bojar et al. 2016), it represents a bias imposed on the model. We therefore experimented with a novel approach, using a neural network that would effectively learn the reward directly. We designed a *discriminator* that simply outputs the probability of an input question being “real" (as opposed to generated by a model). This approach is similar to that used in GANs (Goodfellow et al. 2014), although we initially used a fixed, pre-trained discriminator.

As discussed in Section 3.6, it is problematic if the evaluation metric takes a similar form to the generator itself. The use of a learned discriminator is therefore appealing as it offers an alternative evaluation metric. Theoretically, a properly designed, trained and regularised discriminator would provide a single score representing the quality of a generated question across all criteria.

5.3.1 Architecture

We used an architecture based on a modified QANet as shown in Figure 5.2, replacing the output layers of the model to produce a single probability. Since the discriminator is also able

[2]We used 200 for our experiments.

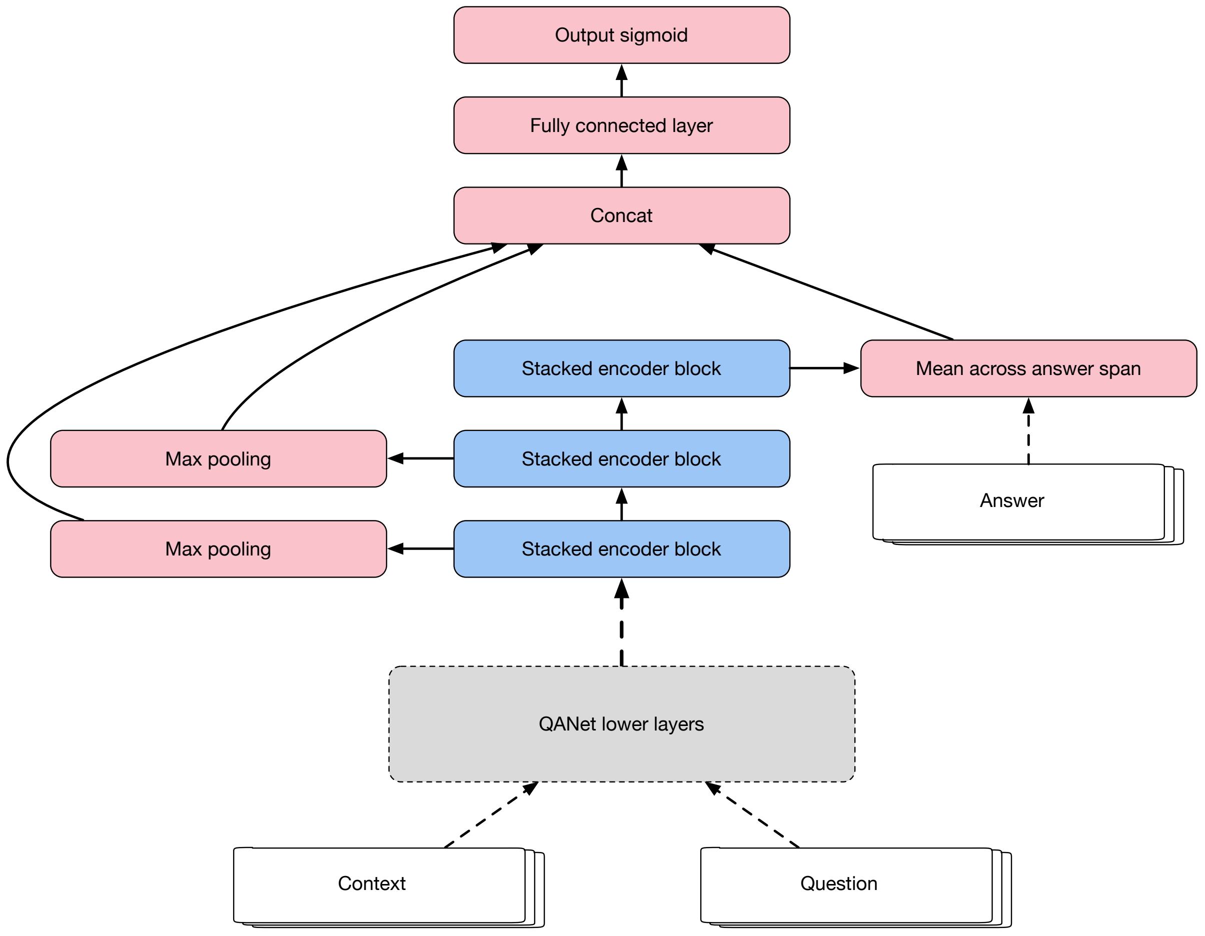


Figure 5.2: Discriminator architecture diagram.

to consider a full context-question-answer triple as input (as opposed to a context-question pair for the QA task), we fused this information in the output layers.

Specifically, we applied max pooling over time to the output of the first two encoders, and we took the mean of the outputs of the third encoder that formed part of the answer span. These three reduced encodings were concatenated, a 64 unit hidden layer with ReLU activation applied, and the output passed through a single unit sigmoid output layer. This architecture should allow the model to consider properties of the generated sequence as a whole, its similarity to the context document, and its similarity to the answer span.

5.3.2 Pre-training results

One advantage of using a model based on an existing architecture is that we can potentially make use of *transfer learning*. Neural networks are powerful as they are able to learn feature representations rather than requiring these features be engineered, with layers in the network closer to the input representing increasingly low level features. These low level features are often independent of the task being solved (Y. Bengio 2011). We can therefore use the parameters for these layers learned for one task as initialisation for another task. Since we have already trained a QANet model on a QA task, we experiment with using the parameters of all layers except the output as initialisation for the discriminator. The model is then *fine tuned* on the specific task of discriminating between generated and ground truth questions.

We pre-trained the discriminator using questions generated by our model, splitting our training set into 80% training and 20% development. The accuracies on our development set are shown in Table 5.3. A breakdown of these predictions is shown in Figure 5.3.

Initialisation	Accuracy (0.5 threshold)	Accuracy (0.2 threshold)
Random	76.5	75.0
QANet	69.5	79.0

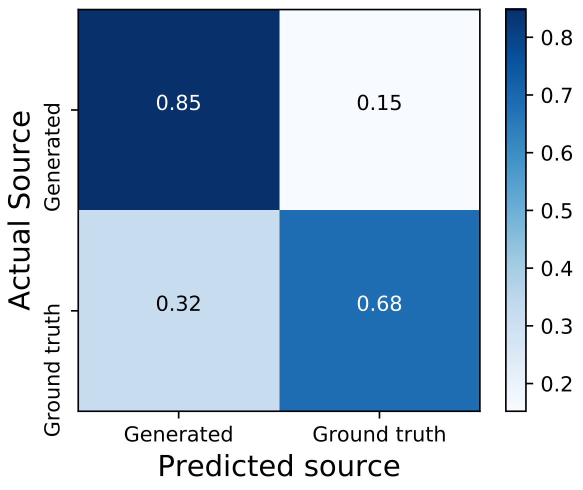
Table 5.3: Discriminator accuracies.

While the accuracies for the two initialisation methods are comparable using a threshold of 0.5, the model initialised with QANet converged in roughly 60% the time of the randomly initialised model.

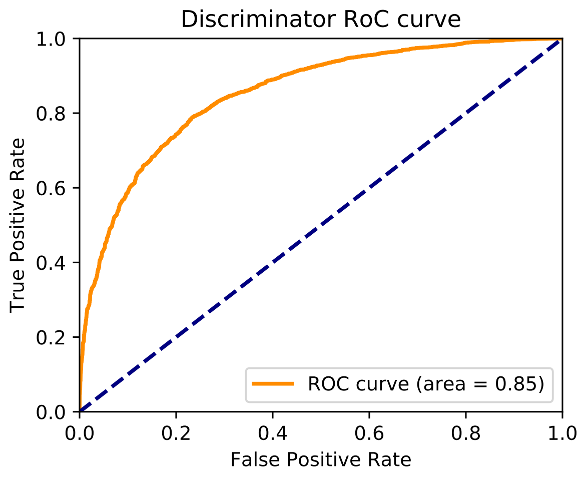
The confusion matrices (Figures 5.3 and 5.4) reveals that the discriminator initiated with QANet reaches 79% accuracy by learning to classify almost all generated questions as being fake, at the expense of misclassifying some of the ground truth examples. The RoC curve suggests that the errors may be better balanced by selecting a decision threshold around 0.2, and that the discriminator initialised using QANet is a more robust estimator, with a higher *area under curve* (AUC). A threshold of 0.2 does indeed improve the accuracy of the QANet initialised model, but not that of the random initialised model.

We note that although the accuracy gives some indication of the performance of the model, we use the unthresholded probability as the reward, and so the choice of threshold does not affect training with a discriminator objective. Since the model initialised with QANet is more robust, we use it to generate rewards for the policy gradient training phase.

The generator is then trained in the same way as for the QA and LM rewards. We found it

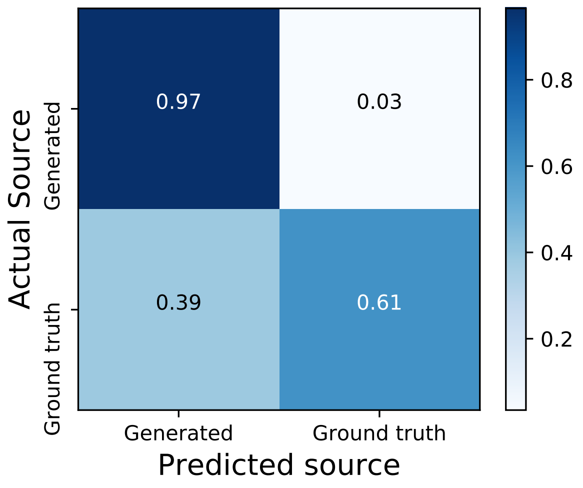


(a) Confusion matrix for discriminator, with a threshold of 0.5

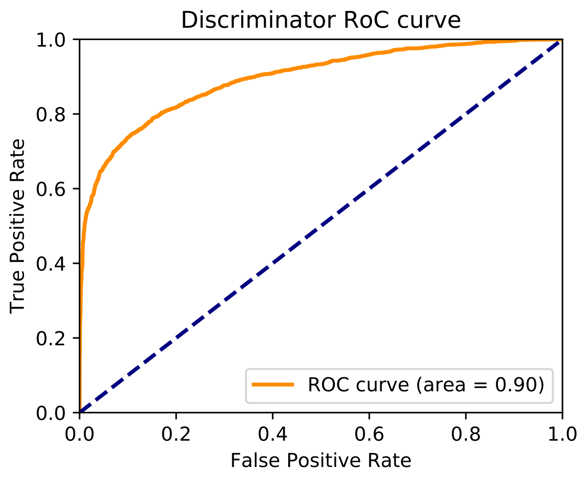


(b) RoC curve for discriminator

Figure 5.3: Pre-trained discriminator performance



(a) Confusion matrix for discriminator, initialised with QANet, with a threshold of 0.5



(b) RoC curve for discriminator

Figure 5.4: Pre-trained discriminator performance, initialised with QANet

was necessary to apply the same whitening procedure to the discriminator rewards to achieve stable training.

5.4 Adversarial training

Following Goodfellow et al. (2014), we experimented with jointly training both the question generator and the discriminator. Both the question generator and discriminator were pre-trained. The discriminator reward was applied to the generator as before, using REINFORCE. After each training step, the discriminator was also updated by randomly selecting either the ground truth or generated question for each context-answer pair in that minibatch.

5.5 Use as an evaluation metric

Initial experiments with an adversarial objective revealed a bug in the generator. The vocabulary was being generated using only the *contexts* from the training set, the top 2000 words of which does not include a question mark. The generator was therefore unable to produce a question mark at the end of generated questions, but the questions otherwise looked reasonable, and were evaluated as competitive by all automatic metrics. However, the jointly trained discriminator was quickly able to learn that generated questions never contained question marks, reaching almost 100% accuracy within a few hundred steps.

This suggests a possible use of the discriminator both as an automated debugging tool for some models and as an alternative evaluation metric. It also suggests that our existing evaluation metrics are potentially flawed and not robust. A QA system should perhaps be trained not just to find the most likely answer, but also to determine whether an input question should be answered at all. Similarly, it raises questions about our choice of language model as a fluency metric - theoretically, a sequence without a question mark at the end should have almost zero probability[3] of occurring, and the perplexity of such a sequence under our language model should reflect this.

A model like the discriminator, that is not biased by human intuitions about the task, also has a major disadvantage: it is hard to interpret what features of a sequence it is using to calculate the score. It may be that the discriminator learns to find a quirk of the output, as in the example above, or that some very basic feature is actually a good predictor, such as sequence length[4].

[3]There are some questions in SQuAD that are stated as tasks, rather than questions.

[4]The correlation between discriminator score and sequence length is in fact very low (0.009), but it would

As noted in Section 3.6, a discriminator trained on a competitive model assigned surprisingly high scores to the model without a shortlist vocabulary, despite these generated questions being almost entirely unstructured. This is further evidence that our discriminator is not in fact assessing the overall quality of generated questions, but instead has identified some characteristic that is structurally different for generated questions and has learned to exploit this.

5.6 Evaluation

Table 5.4 shows the automatic metrics for the models fine tuned on various rewards. We additionally experimented with using a weighted combination of QA, LM and discriminator rewards, but were not able to achieve stable training - this may be possible with a different choice of hyperparameters.

Features				**Metrics**						
QA reward	*LM reward*	*Discriminator reward*	*Adversarial discriminator*	*F1*	*NLL*	*BLEU*	*SOWE*	*QA*	*LM*	*Disc*
Baseline				20.1	-	9.4	85.3	27.8	1328	20.7
No fine tuning				**47.2**	39.6	**14.8**	**91.2**	66.2	64.7	5.0
-	✓	-	-	44.8	38.9	12.9	90.9	62.5	51.3	6.5
✓	-	-	-	39.3	42.3	9.3	87.7	**70.1**	292	10.4
✓	✓	-	-	45.7	39.1	12.2	90.9	68.2	**48.4**	7.9
-	-	✓	-	45.6	**38.8**	13.0	91.0	64.1	55.3	7.5
-	-	✓	✓	44.8	46.0	12.1	91.0	63.7	65.2	**15.8**
✓	✓	✓	✓	45.1	40.6	12.4	90.9	67.5	58.0	15.0
Ground truth				-	-	-	-	71.2	101.5	-

Table 5.4: Automatic evaluation metrics evaluated on models fine tuned using various rewards. The bottom row shows the scores attained by our QA and language models on ground truth questions. The best scores for each metric are highlighted in bold.

be difficult to determine which features the discriminator score *does* correlate with, without first enumerating all possible features.

Training on a QA or LM reward improves these metrics compared to the un-tuned model, but at the cost of a reduction in the other scores. The highest LM score is in fact achieved by training on a weighted sum of both the QA and LM rewards, implying that this joint objective improves stability.

Fine tuning on a reward generated by a pre-trained discriminator does not improve performance, though it does not significantly deteriorate.

Adversarial training using only a discriminator reward, where the discriminator is also updated, leads to the largest improvement in discriminator score, but otherwise leads to small reductions in the other metrics. It is important to note that training is stable, meaning that neither generator nor discriminator find an easy way to optimise their rewards by finding a weakness in the other models architecture.

Including the QA and LM rewards in this adversarial training leads to improved QA, LM and discriminator scores compared to the untuned model.

The examples given in Tables 5.5 and 5.6 demonstrate that the models tuned on either QA or LM rewards have indeed increased these rewards at the cost of the other; the LM model is more fluent but less specific, while the QA model often no longer forms complete questions but exploits the behaviour of the QA model by simply quoting the phrase nearest the answer.

In the examples given, the model tuned adversarially with all three rewards appears to be able to generate more natural questions.

We also note that for all choices of fine tuning reward, the BLEU similarity metric decreased. This is to be expected, as the process of fine tuning using policy gradients decouples the objective from the training data, and the model no longer learns to replicate the ground truth questions.

Context
[...] the t. t. tsui gallery of chinese art opened in 1991 , displaying a representative collection of the v & as approximately 16,000 objects from china , dating from the 4th millennium bc to the present day . though the majority of art works on display date from the **ming and qing** dynasties , there are exquisite examples of objects dating from the tang dynasty and earlier periods . notably , a metre-high bronze head of the buddha dated to c.750 ad and one of the oldest items a 2,000-year-old jade horse head from a burial , other sculptures include life-size tomb guardians . [...]
Answer
ming and qing
Ground Truth Question
most of the chinese works of art in the far eastern collections date from which two dynasties ?
No fine tuning
the majority of art works on display comes from whom ?
LM reward
who are the majority of the majority of art works on display ?
QA reward
the majority of art works on display date from ?
LM and QA reward
who were the majority of art works on display ?
Discriminator reward
the majority of all art works on display ?
Discriminator reward, adversarial discriminator
in what dynasties did the majority of art works on display 's art begin ?
LM, QA and discriminator reward, adversarial discriminator
the majority of art works on display are which dynasties ?

Table 5.5: Example generated questions for various rewards, showing the models learn to optimise one reward at the cost of others.

Context
before the actual research explicitly devoted to the complexity of algorithmic problems started off , numerous foundations were laid out by various researchers . most influential among these was the definition of turing machines by alan turing in **1936** , which turned out to be a very robust and flexible simplification of a computer .
Answer
1936
Ground Truth Question
in what year was the alan turing 's definitional model of a computing device received ?
No fine tuning
in what year did alan turing ?
LM reward
in what year was turing machines ?
QA reward
alan turing in ?
LM and QA reward
in what year did alan ?
Discriminator reward
in what year did alan live ?
Discriminator reward, adversarial discriminator
where did alan turing definition ?
LM, QA and discriminator reward, adversarial discriminator
in what year was the definition of turing machines ?

Table 5.6: Example generated questions for various rewards. The model trained on a QA reward has learned to simply point at the answer and exploit the QA model.

Chapter 6

Summary of Results and Human Evaluation

6.1 Results

Model	Metrics						
	F1	*NLL*	*BLEU*	*SOWE*	*QA*	*LM*	*DISC*
Baseline	20.1	-	9.4	85.3	27.8	1328	20.7
ACE	**47.6**	**37.9**	15.0	91.3	65.9	69.6	9.5
ACE +set vocab +GloVe shortlist	47.4	38.2	**16.3**	**91.8**	**69.0**	91.1	9.3
ACE +latent training +GloVe shortlist	47.2	39.6	14.8	91.2	66.2	64.7	5.0
+ LM and QA reward	45.7	39.1	12.2	90.9	68.2	**48.4**	7.9
+ LM, QA and discriminator reward, adversarial discriminator	45.1	40.6	12.4	90.9	67.5	58.0	**15.0**
QA	-	-	-	-	71.2	-	-
LM	-	-	-	-	-	101.5	-

Table 6.1: Automatic evaluation metrics. Bottom rows show scores for ground truth questions

Table 6.1 shows the automatic metrics for a subset of the models we have investigated. All models outperform the baseline in each metric, except for the discriminator score - as discussed in Section 5.5, this measure does not in general seem to be a good indicator of overall question quality.

Using a shortlist vocabulary constructed from the top 2000 words in GloVe, and using a set-based vocabulary for the copy mechanism leads to an improvement in BLEU score over the model with advanced condition encoding from Yuan et al. (2017). Both of these modifications can be viewed as removing biases from the model, by choosing a more generalised shortlist vocabulary and removing constraints on the switch network, allowing the model to learn the optimal behaviour itself.

Further fine tuning using policy gradients does lead to improvements in the metrics being optimised, but may come at a cost to the other metrics. Tuning on two or more metrics does lead to a balanced improvement in those metrics, but this relies on careful choices of hyperparameters.

Tuning the model using an adversarial discriminator that is also updated leads to stable training, but does not lead to improvements in the automatic metrics being considered.

Model	**Source**	**BLEU**
PCFG-Trans	Zhou et al. (2018)	9.31
Baseline	Our result (dev)	9.4
Baseline	Our result (test)	9.6
Seq2Seq +attn	Zhou et al. (2018)	3.06
Seq2Seq +attn	Our result (dev)	4.0
Seq2Seq +attn	Our result (test)	3.8
Advanced Condition Encoding	Yuan et al. (2017)	10.2
Advanced Condition Encoding	Our result (dev)	14.3
Advanced Condition Encoding	Our result (test)	15.0
NQG++	Zhou et al. (2018)	13.3
M2S +cp	Song et al. (2018)	12.6
QG +F +GAE	Kumar et al. (2018b)	13.9
Dico-QG	Gao et al. (2018)	16.4
Our best model	Our result (dev)	15.5
Our best model	Our result (test)	16.3

Table 6.2: Comparison of BLEU scores against other published results

Table 6.2 shows a comparison of our results against other published results. Unfortunately, there is not yet a standard choice of development and test set splits, and so these results are not strictly comparable. Our best performing model is nonetheless competitive, with our BLEU score of 16.3 very close to the best published result of 16.4. We note that our BLEU score is higher on our test set than development, implying that the test set is perhaps easier.

6.2 Human evaluation

Model	Fluency	Relevance	BLEU
Baseline	2.51	1.85	9.4
ACE	3.34	3.12	15.0
ACE +Set +Glove	**3.51**	**3.42**	**16.3**
ACE +Latent +Glove	3.37	3.17	14.8
ACE +QA +LM	3.05	2.75	12.2
ACE +Adver. disc	2.89	2.82	12.4
Ground Truth	4.67	4.72	-

Table 6.3: Summary of human evaluation of selected models

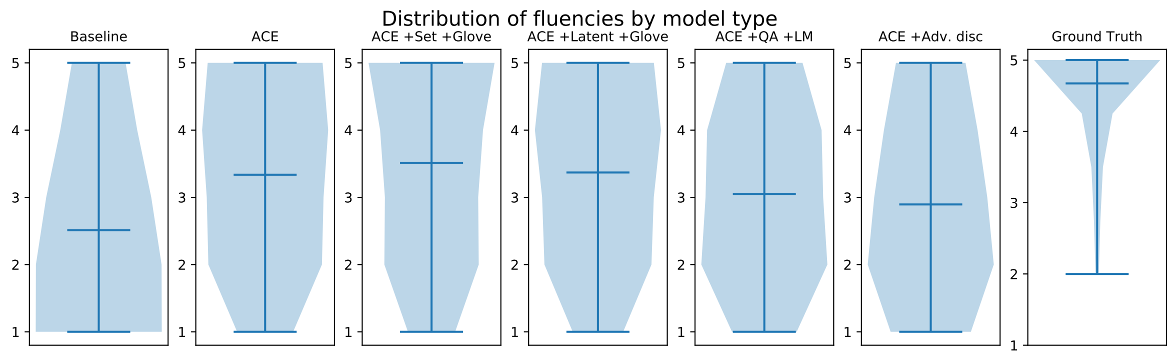
Table 6.3 shows a comparison of the models evaluated by human scorers, along with their respective BLEU scores. Three workers evaluated a total of 300 questions each.

Overall, the ranking of the models by human scores is almost the same as for automatic scores - while the instance level BLEU may not be a perfect measure of the quality of a question, it seems that it is sufficient for our purposes. In particular, all neural models tested outperformed the baseline by a considerable margin, but there is still a significant gap to the ground truth questions.

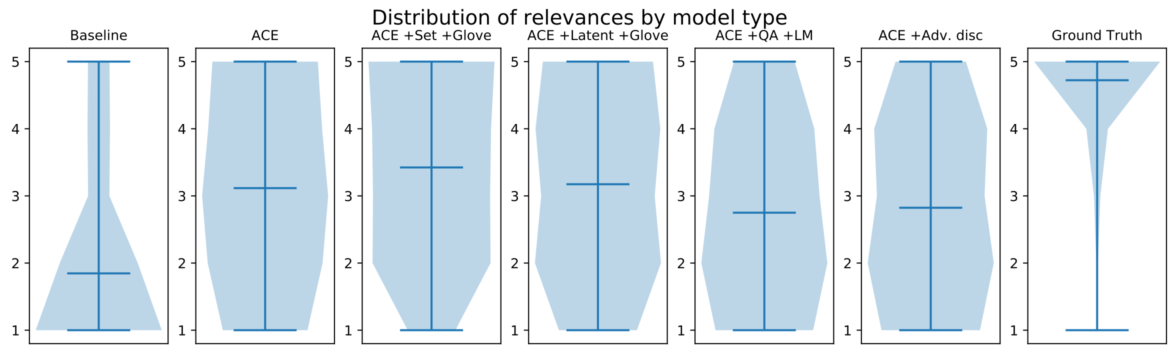
The fluency scores for the models are all higher than the relevancy scores - the models are better at generating coherent language than identify the relevant parts of the context.

Interestingly, the ground truth data is not always assigned a perfect score by the human workers. As shown in Section 2.1.1, some of the crowdsourced questions are misspelled or difficult to answer.

Figure 6.1 shows the distribution of scores for each model, while Figure 6.2 shows the fraction of questions achieving the top score of 5 for each model. The variance in the quality of the questions generated by the neural models seems to be comparable, and none of the models have learned a “riskier" approach that produces more high quality questions at the cost of more complete failures.



(a) Distribution of fluency scores



(b) Distribution of relevance scores

Figure 6.1: Distribution of scores for models

The models that were fine tuned using policy gradients are not rated more highly than the untuned model. We conclude that although policy gradient training improves the performance on the automated metric used as a reward, these higher rewards are not necessarily caused by genuine improvements in question quality.

The mean of the pairwise inter-rater Fleiss' Kappa (L. Fleiss 1971) agreement metrics was 0.45 for fluency, and 0.44 for relevance[1], corresponding to moderate agreement. A score of 0 corresponds to workers assigning random ratings, and a score of 1 to perfect agreement between workers. Since the scores are not strictly different categories, we also calculate the mean pairwise correlation coefficient as 0.95 for fluency and 0.97 for relevance.

Figure 6.4 compares the automatic QA and LM scores against their human counterparts, relevance and fluency. Although in both cases the automatic and human scores are correlated,

[1]Since the source model for each question presented to the worker was randomly selected at each step, not all ratings were for the same model and sample. The number of questions rated by all three workers was therefore very low.

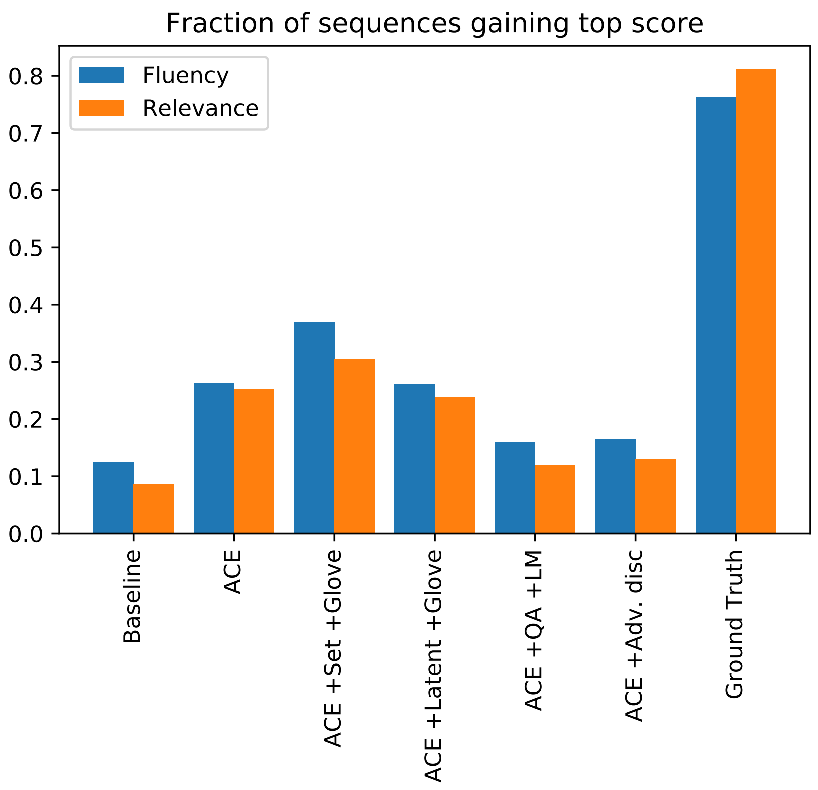


Figure 6.2: Fraction of generation questions achieving scores of 5 in fluency or relevance, by model.

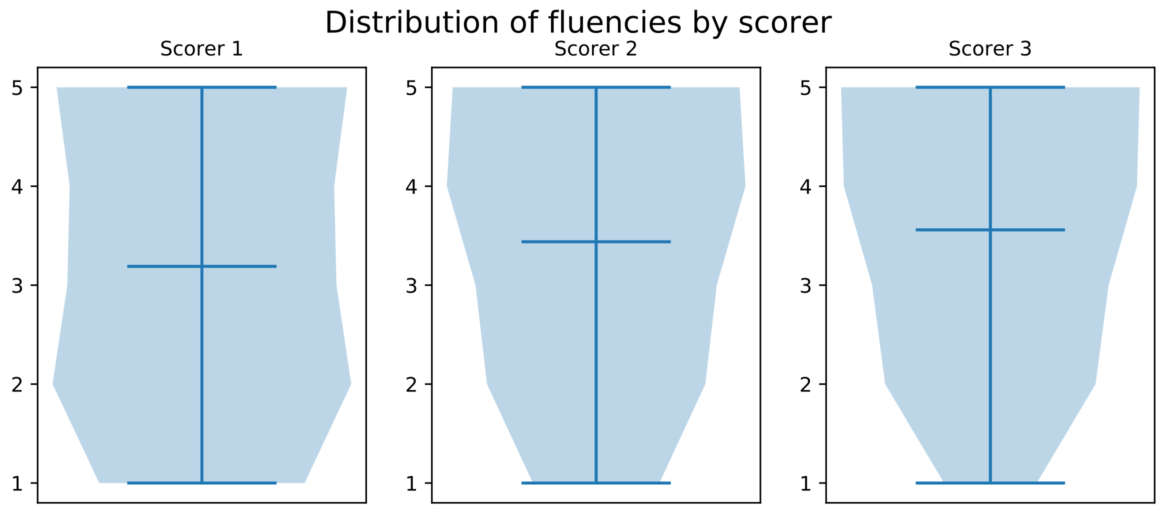
there is still significant noise, and the automatic metrics are far from being ideal proxies for the human scores. While improving the relevance of a question is likely to improve the ability of a QA model to answer it (and similarly for fluency and LM perplexity), we conclude that the inverse is not necessarily true, and tuning on a automatic reward may simply cause the model to learn to exploit the imperfections of that metric.

Adversarial training, where the reward model is able to adapt and theoretically avoid such exploitation, also did not lead to an improvement in question quality.

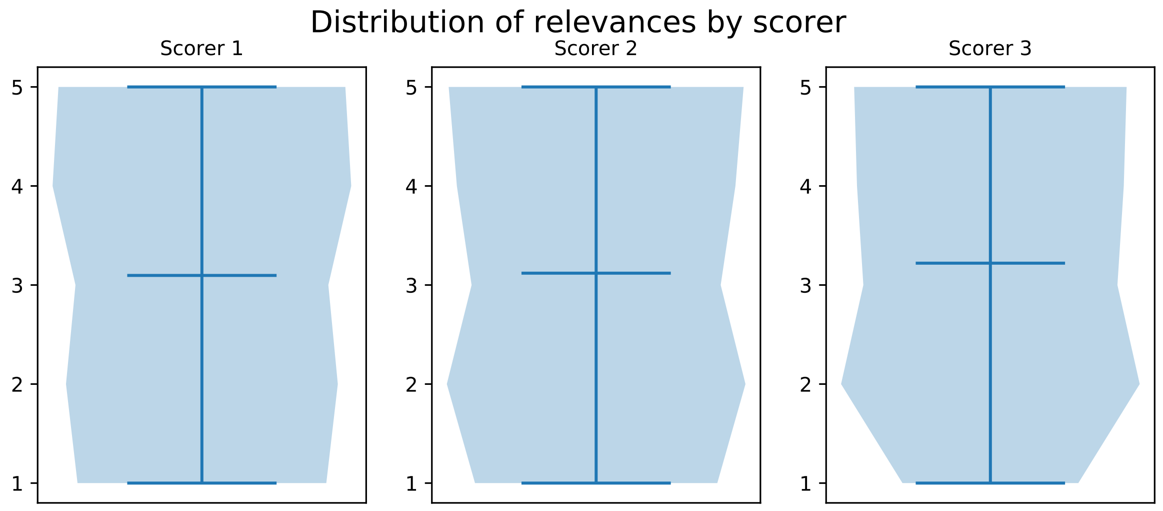
6.3 Error analysis

The confusion matrix in Figure 6.5 shows that the model is often able to correctly select the correct interrogative. There is a strong bias towards selecting “what", which fits with the observation in Figure 2.1 that this is the most common interrogative in the training data by far.

Figure 6.6 shows the distribution of various metrics, split by interrogative. “Who" and “when" questions achieve a higher QA score, since such questions are answered by either dates



(a) Distribution of fluency scores

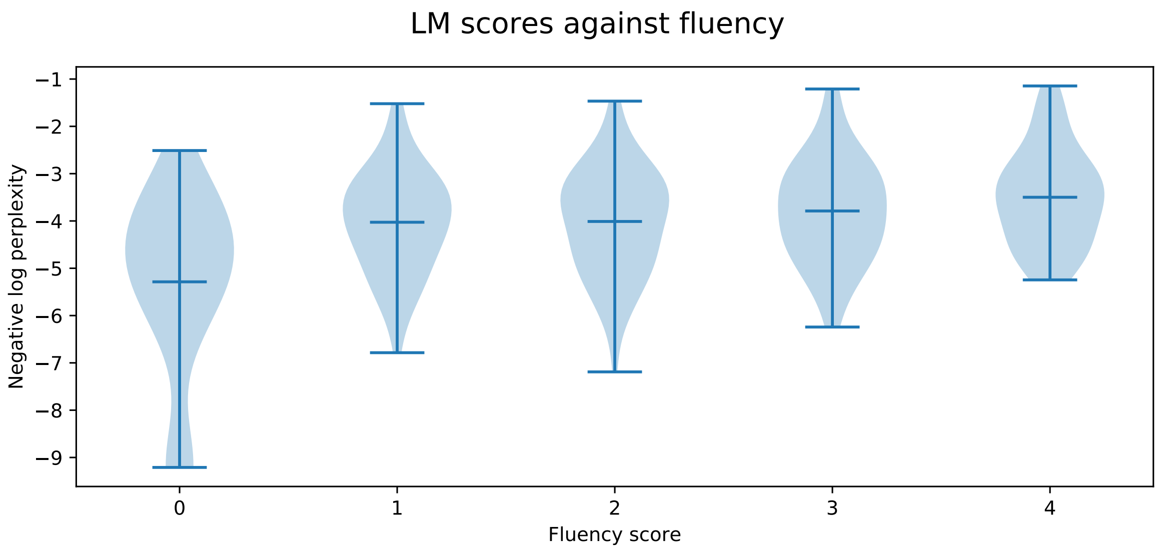


(b) Distribution of relevance scores

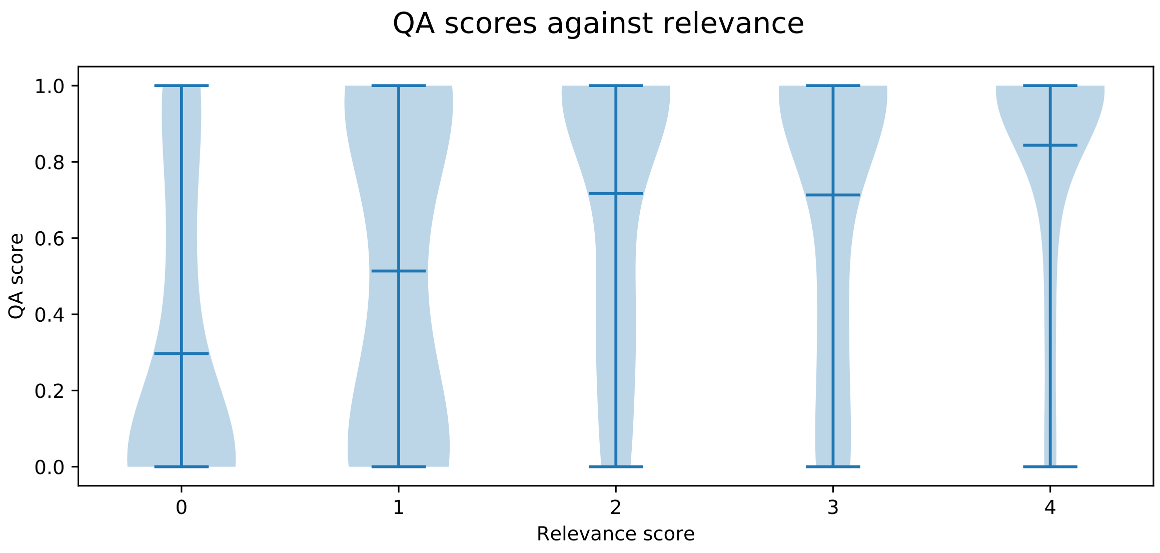
Figure 6.3: Distribution of scores assigned by workers

or named entities, which are easier to identify within a context document. “Which" questions achieve a poor (high) language model perplexity, since they are a more general class of question and are likely to be longer and involve more complex language.

Qualitatively, the generated questions are often close to being good quality, but the model often misses out one or more words that are necessary for the sequence to make sense. On the other hand, the model occasionally degrades into the familiar RNN decoding noise, entering a circular loop and repeating the same phrase repeatedly. The length penalty discussed in



(a) LM negative log perplexities against human fluency scores



(b) QA scores against human relevance scores

Figure 6.4: Comparison of automatic and human metrics

Section 2.5.2 can be modified to balance both of these types of errors, but this presents another hyperparameter to be optimised. It is possible that a variant of the coverage mechanism (Tu et al. 2016) might be used to reduce these types of errors.

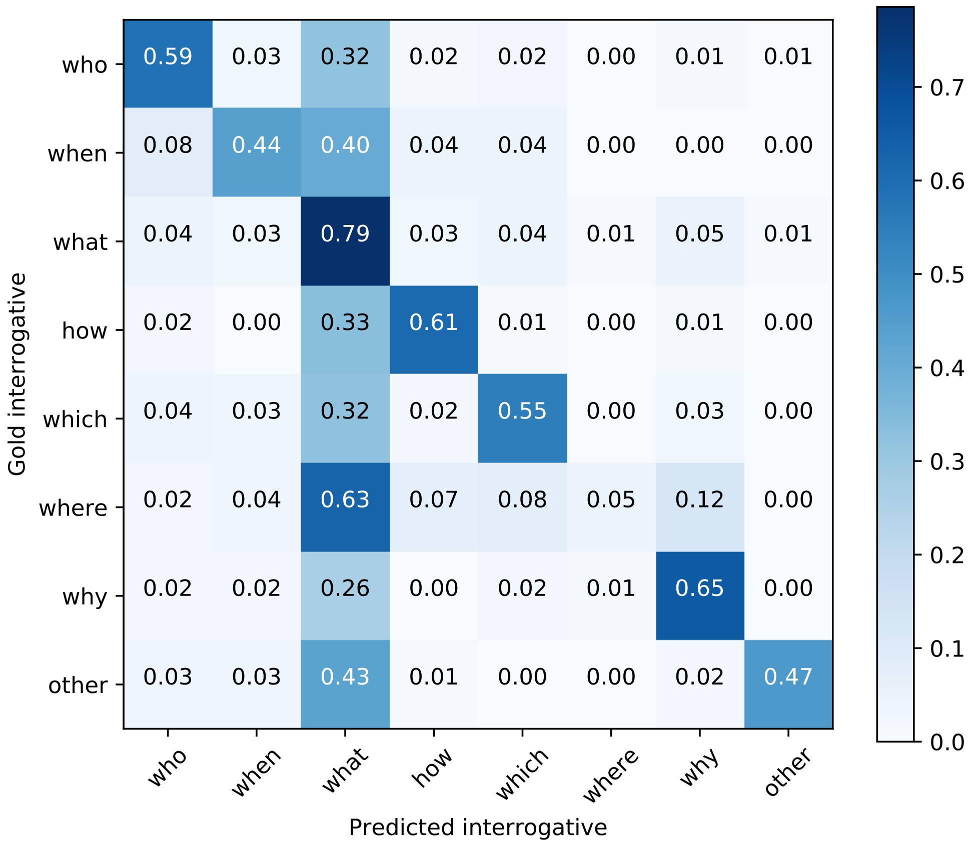


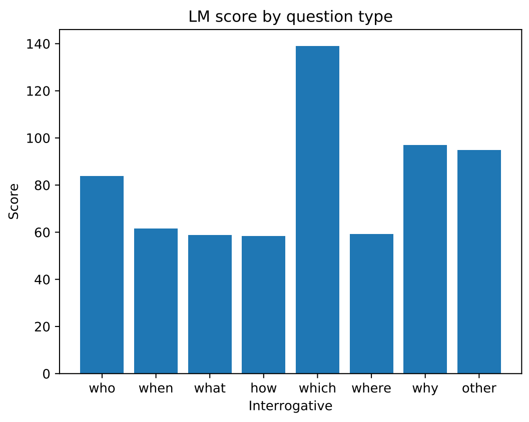
Figure 6.5: Confusion matrix of interrogative words between ground truth and generated questions for the uncropped model with ACE.

6.4 Novel metrics

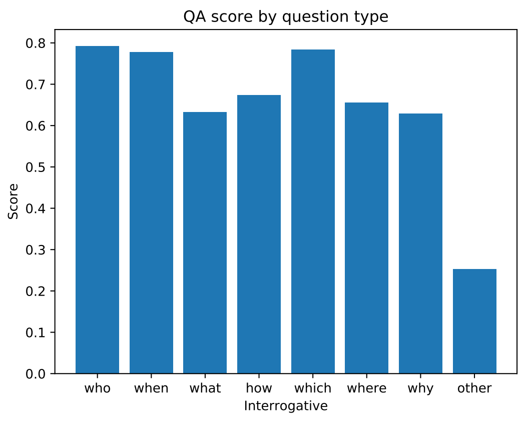
We experimented with training a discriminator to estimate the similarity of a generated question to the ground truth data. Figure 6.7 shows this discriminator score plotted against the human relevance scores. With a correlation of -0.158, it can be seen that the discriminator is not a good predictor of question quality.

Figure 6.8 shows our experimental SOWE metric against human relevance scores. We experimented with use the cosine similarity between the sums of word embeddings between ground truth and generated questions, based on observations by White et al. (2018) that this captures significant semantic information about a sentence.

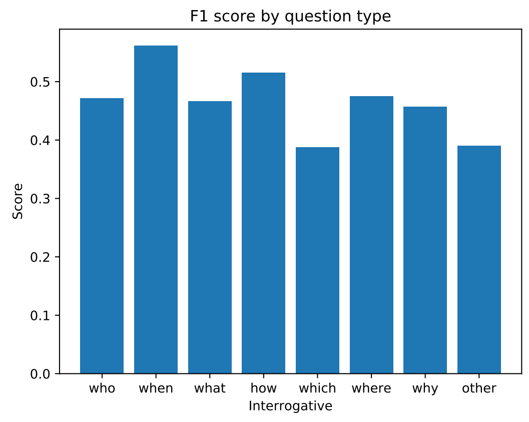
We find there is a correlation of 0.336 between the human relevance score and the SOWE



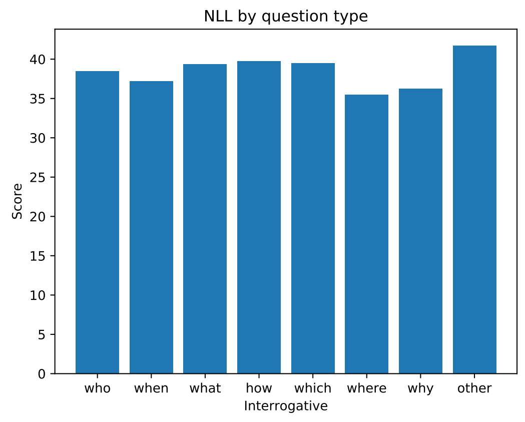
(a) LM scores split by question type



(b) QA scores split by question type



(c) F1 scores split by question type



(d) NLL split by question type

Figure 6.6: Scores split by question type for the uncropped model with ACE.

metric. While it appears to be able to identify completely irrelevant sentences, a high SOWE score does not seem to be a good predictor of the relevance of a question. The use of embedding similarity as an evaluation metric has a number of desirable properties, being fast to calculate and insensitive to particular word choice, but this particular formulation is not sufficiently accurate as to be useful.

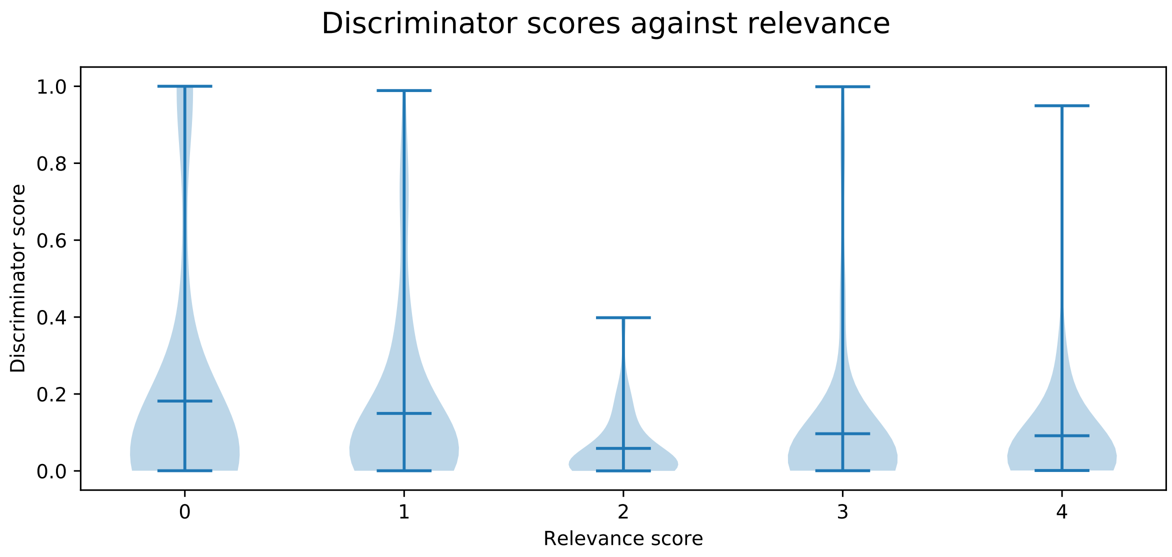


Figure 6.7: Discriminator score against human relevance scores

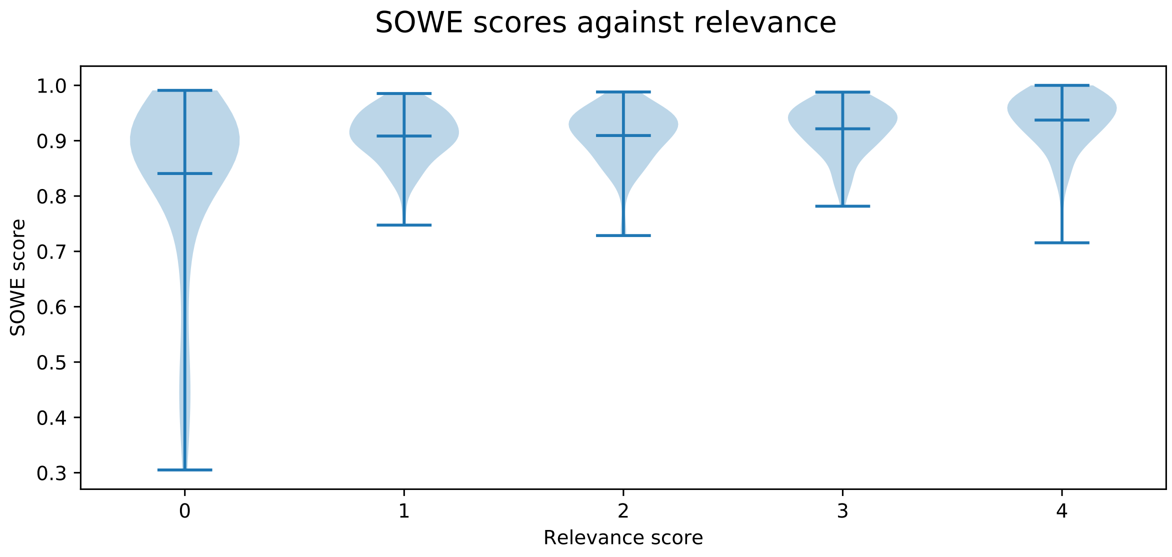
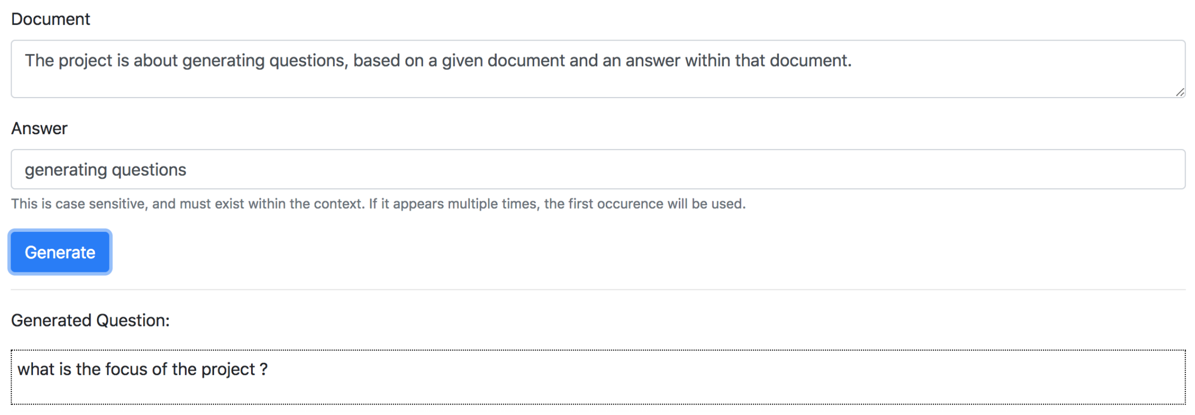


Figure 6.8: SOWE metric against human relevance scores

Chapter 7

Conclusion

7.1 Summary


Document

The project is about generating questions, based on a given document and an answer within that document.

Answer

generating questions

This is case sensitive, and must exist within the context. If it appears multiple times, the first occurence will be used.

Generated Question:

what is the focus of the project ?

Generating natural language questions from a document and answer is a challenging task, requiring both understanding of the input text and the ability to generate convincing and relevant language.

This work aimed to evaluate and examine the current models in the field, and we answered four research questions to this effect.

We found that we were able to generate natural language questions about a document, and that these outperformed baseline non-neural models when assessed by human evaluators.

We found that removing constraints and biases from the model improved its ability to generate convincing language, and showed that the pointer network is a relatively robust method for generating language.

We showed that policy gradient and adversarial techniques can be used to increase the scores attained by generated questions.

The use of external models as proxies for the fluency and relevance of a question is useful, but we found that these are not perfect replacements, with only a moderate level of agreement between human and machine.

7.2 Future work

Following the observations made by White et al. (2018), we briefly experimented with training a model on a loss function that measured the embedding similarity between predicted and ground truth tokens, to reduce the reliance on the training data and reduce exposure bias.

As we noted in Chapter 1, a machine learning system can be improved by considering three main aspects: those of the model design, the training scheme, and the inference method used for sampling from the model. While we considered the first two, we did not investigate possible approaches for improving decoding from the model. Diverse beam search (Vijayakumar et al. 2016) has been shown to improve diversity of sampled sequences, by tackling the propensity of beam search to generate sequences that are strongly biased by the first few tokens sampled. Regarding the non-differentiability of beam search, Goyal et al. (2017) showed that by using a continuous approximation to the hard argmax function, it is possible to perform beam search in a differentiable manner. This might then allow the parameters of the model to be updated directly and without the stability issues common to policy gradient training.

The decision to train using rewards generated by external models was, to an extent, forced on us by the lack of diverse ground truth data. Our repurposing of a QA dataset means there is only one question per answer, and it would be valuable to tackle this issue directly by generating a dedicated question generation dataset.

While we found that a reward based on a neural model is not sufficient to improve the quality of generated questions, the system could be used in an active learning context. Deployed as a revision aid or automatic tutor, feedback from real users could be used to create a dedicated question learning dataset and iteratively improve the model by training on this feedback.

Appendix A

Hyperparameter values

Parameter	**Value**
Learning rate	2×10^{-4}
Embedding dimension	200
RNN hidden units	768
Shortlist vocab size	2000
Dropout rate	0.3
Beam width	32
Entropy loss weight	0.01
Suppression loss weight	0.01

Table A.1: Hyperparameter values used for all model types

Parameter	**Value**
Vocab size	$20,000$
Embedding dimension	200
RNN hidden units	384
Dropout rate	0.3

Table A.2: Hyperparameter values used for language model

Model type	**Batch size**
Uncropped	32
Cropped	64

Table A.3: Model specific batch sizes

Model type	**LM weight**	**QA weight**	**Discriminator weight**
LM	0.1	-	-
QA	-	1.0	-
LM+QA	0.25	0.5	-
Disc	-	-	1.0
Disc+Adversarial	-	-	1.0
LM+QA+Disc	0.25	0.5	0.5

Table A.4: Model specific reward weights

Bibliography

Agarwal, Manish, Rakshit Shah, and Prashanth Mannem (2011). "Automatic question generation using discourse cues". In: *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications* June, pp. 1–9.

Ali, Husam, Yllias Chali, and S. Hasan (2010). "Automation of Question Generation from Sentences". In: *Proceedings of QG2010: The Third Workshop on Question Generation*, pp. 58–67.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton (2016). "Layer Normalization". In: ISSN: 1607.06450. arXiv: 1607.06450.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural Machine Translation by Jointly Learning to Align and Translate". In: ISSN: 0147-006X. arXiv: 1409.0473.

Bahuleyan, Hareesh et al. (2017). "Variational Attention for Sequence-to-Sequence Models". In: arXiv: 1712.08207.

Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun (2018). "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". In: arXiv: 1803.01271.

Banerjee, Satanjeev and Alon Lavie (2005). "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments". In: *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization* June, pp. 65–72. ISSN: 09226567.

Bengio, Samy et al. (2015). "Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks". In: ISSN: 1941-6016. arXiv: 1506.03099.

Bengio, Yoshua (2011). “Deep Learning of Representations for Unsupervised and Transfer Learning”. In: *JMLR: Workshop and Conference Proceedings*. Vol. 7, pp. 1–20. ISBN: 9780971977778. arXiv: `1606.09549`.

Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent (2000). “A neural probabilistic language model”. In: *International Conference on Advances in Neural Information Processing Systems (NIPS)*. Vol. 3, pp. 932–938.

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi (1994). “Learning Long-Term Dependencies with Gradient Descent is Difficult”. In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. ISSN: 19410093. arXiv: `arXiv:1211.5063v2`.

Bojar, Ondrej et al. (2016). “Findings of the 2016 Conference on Machine Translation (WMT16)”. In: *WMT-2016*. Vol. 2, pp. 131–198.

Chaganty, Arun Tejasvi, Stephen Mussman, and Percy Liang (2018). “The price of debiasing automatic metrics in natural language evaluation”. In: arXiv: `1807.02202`.

Chali, Yllias and Sina Golestanirad (2016). “Ranking Automatically Generated Questions Using Common Human Queries”. In: *Proceedings of the 9th International Natural Language Generation conference*, pp. 217–221.

Che, Tong et al. (2017). *Maximum-Likelihood Augmented Discrete Generative Adversarial Networks*. Tech. rep. arXiv: `1702.07983`.

Cheng, Jianpeng and Mirella Lapata (2016). “Neural Summarization by Extracting Sentences and Words”. In: ISSN: 9781941643723. arXiv: `1603.07252`.

Cho, Kyunghyun et al. (2014). “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: ISSN: 09205691. arXiv: `1406.1078`.

Danon, Guy and Mark Last (2017). “A Syntactic Approach to Domain-Specific Automatic Question Generation”. In: arXiv: `1712.09827`.

Deng, J. et al. (2009). “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255.

Du, Xinya and Claire Cardie (2018). “Harvesting Paragraph-Level Question-Answer Pairs from Wikipedia”. In: arXiv: `1805.05942`.

Du, Xinya, Junru Shao, and Claire Cardie (2017). “Learning to Ask: Neural Question Generation for Reading Comprehension”. In: pp. 1342–1352. ISSN: 10495258. arXiv: `1705.00106`.

Fan, Zhihao et al. (2018). *A Reinforcement Learning Framework for Natural Question Generation using Bi-discriminators*. Tech. rep.

Firth, J. R. (1957). “A synopsis of linguistic theory 1930-55.” In: 1952-59, pp. 1–32.

Gal, Yarin and Zoubin Ghahramani (2015). “A Theoretically Grounded Application of Dropout in Recurrent Neural Networks”. In: arXiv: `1512.05287`.

Gao, Yifan et al. (2018). “Difficulty Controllable Question Generation for Reading Comprehension”. In: arXiv: `1807.03586`.

Gehring, Jonas et al. (2017). “Convolutional Sequence to Sequence Learning”. In: ISSN: 1938-7228. arXiv: `1705.03122`.

Glorot, Xavier and Yoshua Bengio (2010). “Understanding the difficulty of training deep feedforward neural networks”. In: *PMLR* 9, pp. 249–256. ISSN: 15324435. arXiv: `arXiv:1011.1669v3`.

Goodfellow, Ian et al. (2014). “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. ISSN: 10495258. arXiv: `arXiv:1406.2661v1`.

Goyal, Kartik et al. (2017). “A Continuous Relaxation of Beam Search for End-to-end Training of Neural Sequence Models”. In: arXiv: `1708.00111`.

Graves, Alex (2012). “Sequence Transduction with Recurrent Neural Networks”. In: ISSN: 18792782. arXiv: `1211.3711`.

Gu, Jiatao et al. (2016a). “Incorporating Copying Mechanism in Sequence-to-Sequence Learning”. In: arXiv: `1603.06393`.

Gu, Jiatao et al. (2016b). “Incorporating Copying Mechanism in Sequence-to-Sequence Learning”. In: arXiv: `1603.06393`.

Gulcehre, Caglar et al. (2016). “Pointing the Unknown Words”. In: ISSN: 18770509. arXiv: `1603.08148`.

Hasselt, Hado Van et al. (2016). “Learning functions across many orders of magnitudes”. In: *arXiv* Nips, pp. 1–19. ISSN: 10495258. arXiv: `arXiv:1602.07714v1`.

He, Kaiming et al. (2015). “Deep Residual Learning for Image Recognition”. In: *Arxiv.Org* 7.3, pp. 171–180. ISSN: 1664-1078. arXiv: `1512.03385`.

Heilman, Michael (2011). “Automatic factual question generation from text”. In: *Language Technologies Institute School of Computer Science Carnegie Mellon University* 195.

Heilman, Michael and Noah A Smith (2010). “Good question! Statistical ranking for question generation”. In: *NAACL HLT 2010 - Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, pp. 609–617.

Hermann, Karl Moritz et al. (2015). “Teaching Machines to Read and Comprehend”. In: ISSN: 10495258. arXiv: `1506.03340`.

Hill, Felix et al. (2015). “The Goldilocks Principle: Reading Children’s Books with Explicit Memory Representations”. In: ISSN: 18238262. arXiv: `1511.02301`.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long Short-Term Memory”. In: *Neural Computation* 9.8, pp. 1735–1780. ISSN: 08997667. arXiv: `1206.2944`.

Hochreiter, Sepp et al. (2001). “Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies”. In: *A Field Guide to Dynamical Recurrent Networks*. ISBN: 978-0-7803-5369-5. arXiv: `arXiv:1011.1669v3`.

Hopfield, J J (1982). “Neural networks and physical systems with emergent collective computational abilities.” In: *Proceedings of the National Academy of Sciences* 79.8, pp. 2554–2558. ISSN: 0027-8424. arXiv: `arXiv:1411.3159v1`.

Huszár, Ferenc (2015). “How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?” In: arXiv: `1511.05101`.

Indurthi, Sathish et al. (2017). “Generating Natural Language Question-Answer Pairs from a Knowledge Graph Using a RNN Based Question Generation Model”. In: *Eacl* 1, pp. 376–385.

Jean, Sébastien et al. (2014). “On Using Very Large Target Vocabulary for Neural Machine Translation”. In: ISSN: 0147-006X. arXiv: `1412.2007`.

Joshi, Mandar et al. (2017). “TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension”. In: arXiv: `1705.03551`.

Kandasamy, Kirthevasan et al. (2017). “Batch Policy Gradient Methods for Improving Neural Conversation Models”. In: arXiv: `1702.03334`.

Kim, Min Sang (2018). *QANet*. `https://github.com/NLPLearn/QANet`.

Kingma, Diederik and Jimmy Ba (2014). “Adam: A Method for Stochastic Optimization”. In:

Koehn, Philipp and Christof Monz (2006). “Manual and Automatic Evaluation of Machine Translation between European Languages”. In: *Proceedings of the Workshop on Statistical Machine Translation*, pp. 102–121.

Krizhevsky, Alex, Ilya Sutskever, and G Hinton (2012). *ImageNet classification with deep convolutional neural networks*.

Kumar, Vishwajeet, Ganesh Ramakrishnan, and Yuan-Fang Li (2018a). *A Framework for Automatic Question Generation from Text using Deep Reinforcement Learning*. Tech. rep. arXiv: `arXiv:1808.04961v1`.

Kumar, Vishwajeet et al. (2018b). “Automating reading comprehension by generating question and answer pairs”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10939 LNAI, pp. 335–348. ISBN: 9783319930398. arXiv: `1803.03664`.

L. Fleiss, Joseph (1971). “Measuring Nominal Scale Agreement Among Many Raters”. In: 76, pp. 378–.

Labutov, Igor, Sumit Basu, and Lucy Vanderwende (2015). “Deep Questions without Deep Understanding”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 889–898.

LeCun, Yann A., Yoshua Bengio, and Geoffrey E. Hinton (2015). “Deep learning”. In: *Nature* 521.7553, pp. 436–444. ISSN: 0028-0836. arXiv: `arXiv:1312.6184v5`.

LeCun, Yann et al. (1990). “Handwritten Digit Recognition with a Back-Propagation Network”. In: *Advances in Neural Information Processing Systems 2*. Ed. by D. S. Touretzky. Morgan-Kaufmann, pp. 396–404.

Li, Jiwei et al. (2016). “Deep Reinforcement Learning for Dialogue Generation”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1192–1202. ISBN: 9781509061822. arXiv: `1606.01541`.

Little, W. A. (1974). “The existence of persistent states in the brain”. In: *Mathematical Biosciences* 19.1-2, pp. 101–120. ISSN: 00255564. arXiv: `arXiv:1011.1669v3`.

Loper, Edward and Steven Bird (2002). “NLTK: The Natural Language Toolkit”. In: *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics* - 1, pp. 63–70. arXiv: `0205028 [cs]`.

Martin Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org.

Mazidi, Karen and Rodney D Nielsen (2014). “Linguistic considerations in automatic question generation”. In: *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference* 2, pp. 321–326.

Mikolov, Tomas et al. (2013a). “Distributed Representations of Words and Phrases and their Compositionality”. In: arXiv: `1310.4546`.

Mikolov, Tomas et al. (2013b). “Efficient Estimation of Word Representations in Vector Space”. In: *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, pp. 1–12. ISSN: 15324435. arXiv: `arXiv:1301.3781v3`.

Morgan, Nelson and Hervé Bourlard (1989). “Generalization and Parameter Estimation in Feedforward Nets: Some Experiments”. In: *Adv. NIPS 2*, pp. 630–637.

Nallapati, Ramesh, Feifei Zhai, and Bowen Zhou (2016a). “SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents”. In: arXiv: `1611.04230`.

Nallapati, Ramesh et al. (2016b). “Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond”. In: arXiv: `1602.06023`.

Ney, Hermann, Ute Essen, and Reinhard Kneser (1994). *On structuring probabilistic dependences in stochastic language modelling*.

Nguyen, Tri et al. (2016). “MS MARCO: A human generated MAchine reading COmprehension dataset”. In: *CEUR Workshop Proceedings*. Vol. 1773. ISBN: 00010782. arXiv: `1611.09268`.

Norouzi, Mohammad et al. (2016). “Reward Augmented Maximum Likelihood for Neural Structured Prediction”. In: ISSN: 03044149. arXiv: `1609.00150`.

Papineni, Kishore et al. (2002). “BLEU: a method for automatic evaluation of machine translation”. In: *... of the 40Th Annual Meeting on ...* July, pp. 311–318. ISSN: 00134686. arXiv: 1702.00764.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2012). “On the difficulty of training Recurrent Neural Networks”. In: ISSN: 1045-9227. arXiv: 1211.5063.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). “Glove: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. ISBN: 9781937284961. arXiv: 1504.06654.

Peters, Matthew E. et al. (2018). “Deep contextualized word representations”. In: *NAACL*. ISSN: 9781941643327. arXiv: 1802.05365.

Popowich, Fred and Phil Winne (2013). “Generating Natural Language Questions to Support Learning On-Line”. In: *RITA, Revista de Informática Teórica e Aplicada* 20, pp. 105–114.

Rajpurkar, Pranav, Robin Jia, and Percy Liang (2018). “Know What You Don’t Know: Unanswerable Questions for SQuAD”. In: arXiv: 1806.03822.

Rajpurkar, Pranav et al. (2016). “SQuAD: 100,000+ Questions for Machine Comprehension of Text”. In: ISSN: 9781941643327. arXiv: 1606.05250.

Ranzato, Marc’Aurelio et al. (2015). “Sequence Level Training with Recurrent Neural Networks”. In: ISSN: 15537358. arXiv: 1511.06732.

Saeidi, Marzieh et al. (2018). “Interpretation of Natural Language Rules in Conversational Machine Reading”. In: *Empirical Methods in Natural Language Processing (EMNLP)*.

See, Abigail, Peter J. Liu, and Christopher D Manning (2017). “Get To The Point: Summarization with Pointer-Generator Networks”. In: arXiv: 1704.04368.

Selvaraju, Ramprasaath R. et al. (2017). “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *Proceedings of the IEEE International Conference on Computer Vision*. Vol. 2017-Octob, pp. 618–626. ISBN: 9781538610329. arXiv: 1610.02391.

Seo, Minjoon et al. (2016). “Bidirectional Attention Flow for Machine Comprehension”. In: arXiv: 1611.01603.

Serban, Iulian Vlad et al. (2016). “Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus”. In: arXiv: `1603.06807`.

Shi, Xing, Kevin Knight, and Deniz Yuret (2016). “Why Neural Translations are the Right Length”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*, pp. 2278–2282.

Singer, Harry and Dan Donlan (1982). “Active Comprehension: Problem-Solving Schema with Question Generation for Comprehension of Complex Short Stories”. In: *Reading Research Quarterly* 17.2, p. 166. ISSN: 00340553.

Song, Linfeng et al. (2018). “Leveraging Context Information for Natural Question Generation”. In: pp. 569–574.

Srivastava, Nitish et al. (2014). “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15, pp. 1929–1958. ISSN: 15337928. arXiv: `1102.4807`.

Subramanian, Sandeep et al. (2017). “Neural Models for Key Phrase Detection and Question Generation”. In: pp. 1–11. arXiv: `1706.04560`.

Sundermeyer, Martin, Ralf Schl, and Hermann Ney (2012). “LSTM Neural Networks for Language Modeling”. In: *Proc. Interspeech*, pp. 194–197.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). “Sequence to Sequence Learning with Neural Networks”. In: *Nips*, p. 9. ISSN: 09205691. arXiv: `1409.3215`.

Tang, Duyu et al. (2017). “Question Answering and Question Generation as Dual Tasks”. In: arXiv: `1706.02027`.

Trischler, Adam et al. (2016). “NewsQA: A Machine Comprehension Dataset”. In: *Synthesis* 41.10, pp. 1919–1929. arXiv: `1611.09830`.

Tu, Zhaopeng et al. (2016). “Coverage-based Neural Machine Translation”. In: *Arxiv*, pp. 1–19. ISSN: 2307-387X. arXiv: `1601.04811`.

Tuan, Yi-Lin and Hung-yi Lee (2018). *Improving Conditional Sequence Generative Adversarial Networks by Stepwise Evaluation.*

Vaswani, Ashish et al. (2017). “Attention Is All You Need”. In: ISSN: 0140-525X. arXiv: `1706.03762`.

Vijayakumar, Ashwin K et al. (2016). “Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models”. In: arXiv: `1610.02424`.

Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly (2015). “Pointer Networks”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, pp. 2692–2700. ISSN: 10495258. arXiv: `1506.03134`.

Wang, Tong, Xingdi Yuan, and Adam Trischler (2017). “A Joint Model for Question Answering and Question Generation”. In: *Arxiv*. arXiv: `1706.01450`.

Wang, Zhiguo et al. (2016). “Multi-Perspective Context Matching for Machine Comprehension”. In: arXiv: `1612.04211`.

White, Lyndon et al. (2017). “Modelling Sentence Generation from Sum of Word Embedding Vectors as a Mixed Integer Programming Problem”. In: *IEEE International Conference on Data Mining Workshops, ICDMW*. IEEE, pp. 770–777. ISBN: 9781509054725.

— (2018). “Generating bags of words from the sums of their word embeddings”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9623 LNCS. Springer, Cham, pp. 91–102. ISBN: 9783319754765.

Williams, Ronald J. (1992). “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning”. In: *Machine Learning* 8.3, pp. 229–256. ISSN: 15730565.

Williams, Ronald J. and David Zipser (1989). “A Learning Algorithm for Continually Running Fully Recurrent Neural Networks”. In: *Neural Computation* 1.2, pp. 270–280. ISSN: 0899-7667. arXiv: `arXiv:1011.1669v3`.

Wu, Lijun et al. (2017). “Adversarial Neural Machine Translation”. In: arXiv: `1704.06933`.

Wu, Yonghui et al. (2016). “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”. In: *ArXiv e-prints*, pp. 1–23. ISSN: 1609.08144. arXiv: `1609.08144v2`.

Yang, Yi, Scott Wen-tau Yih, and Chris Meek (2015). “WikiQA: A Challenge Dataset for Open-Domain Question Answering”. In: ACL Association for Computational Linguistics.

Yang, Zhen et al. (2017). “Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets”. In: arXiv: `1703.04887`.

Yang, Zhilin et al. (2017). “Semi-Supervised QA with Generative Domain-Adaptive Nets”. In: arXiv: `1702.02206`.

Yu, Adams Wei et al. (2018). “QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension”. In: arXiv: `1804.09541`.

Yu, Lantao et al. (2016). “SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient”. In: ISSN: 21686106. arXiv: `1609.05473`.

Yuan, Xingdi et al. (2017). “Machine Comprehension by Text-to-Text Neural Question Generation”. In: arXiv: `1705.02012`.

Zhang, Xingxing and Mirella Lapata (2017). “Sentence Simplification with Deep Reinforcement Learning”. In: arXiv: `1703.10931`.

Zhou, Qingyu et al. (2018). “Neural question generation from text: A preliminary study”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* Vol. 10619 LNAI. Springer, Cham, pp. 662–671. ISBN: 9783319736174. arXiv: `1704.01792`.