FastQA: A Simple and Efficient Neural Architecture for Question Answering

Dirk Weissenborn **Georg Wiese** **Laura Seiffe**
Language Technology Lab, DFKI
Alt-Moabit 91c
Berlin, Germany
{diwe01,gewi01,lase01}@dfki.de

Abstract

Recent development of large-scale question answering (QA) datasets triggered [illegible] substantial amount of research into end-to-end neural architectures for QA. Increasingly complex systems have been conceived without comparison to a simpler neural baseline system that would justify their complexity. In this work, we propose a simple heuristic [illegible]t guided the development of FastQA, an efficient end-to-end neural model for question answering that is very competitive with existing models. We further demonstrate, that an extended version (FastQAExt) achieves state-of-the-art results on recent benchmark datasets, namely SQuAD, NewsQA and MsMARCO, outperforming most existing models. However, we show that increasing the complexity of FastQA to FastQAExt does not yield any systematic improvements. We argue that the same holds true for most existing systems that are similar to FastQAExt. A manual analysis reveals that our proposed heuristic explains most predictions of our model, which indicates that modeling a simple heuristic is enough to achieve strong performance on extractive QA datasets. The overall strong performance of FastQA puts results of existing, more complex models into perspective.

1 Introduction

Question answering is an important end-user task at the intersection of natural language processing (NLP) and information retrieval (IR). QA systems can bridge the gap between IR-based search engines and sophisticated intelligent assistants that enable a more directed information retrieval process. Such systems aim at finding precisely the piece of information sought by the user instead of documents or snippets containing the answer. This requires an in-depth understanding of the provided query or question. A special form of QA, namely extractive QA, deals with the extraction of a direct answer to a question from a given textual context. It requires jointly modeling a natural language question in conjunction with its provided context.

The creation of large-scale, extractive QA datasets (Rajpurkar et al., 2016; Trischler et al., 2017; Nguyen et al., 2016) sparked research interest into the development of end-to-end neural QA systems. A typical neural architecture consists of an encoder that encodes the question and context into a sequence of n-dimensional representations, an interaction layer which allows for the interaction between question and context, and an answer layer responsible for extracting the answer span within the context. Many models of this type have been proposed very recently (Wang and Jiang, 2017; Yu et al., 2017; Xiong et al., 2017; Seo et al., 2017; Yang et al., 2017; Wang et al., 2017). Each of those describe several innovations for the different layers of the architecture with a special focus on developing powerful interaction layers.

Although a variety of extractive QA systems have been proposed, there is no competitive neural baseline. Most systems were built in what we call a *top-down* process that proposes a complex architecture first and validates design decisions by an ablation study. Most ablation studies, however, remove only a single part of an overall complex architecture and therefore lack comparison to a reasonable neural baseline. This gap raises the question whether the complexity of current systems is justified solely by their empirical results.

In this work, we propose a heuristic that serves as a simple baseline for the extractive QA task. It selects answer spans that match the expected answer type and that are close to important question words. Expected answer types are not limited and can be inferred directly from the question with varying granularity. Based on this context/type matching heuristic we derive FastQA, a neural network architecture that is efficient wrt. space- and time complexity as well as performance. Its architecture is the result of an incremental e[illegible]nsion process (*bottom-up*) which starts from a very basic model. Individual extensions are created by necessity with simplicity in mind and empirically validated. Crucially, we do not make use of a complex interaction layer but model interaction between question and context through computable features on the word level. FastQA's strong performance questions the necessity of additional complexity, especially in the interaction layer, that is exhibited by recently developed models. We address this issue by evaluating the impact of an additional interaction layer in more detail. In addition, we provide a qualitative analysis that gives insights into the capabilities of FastQA. The analysis indicates that our system indeed mostly learns our context/-type matching heuristic. Finally, the contributions of this work are the following:

- definition of an efficient neural QA model (FastQA) based on a simple heuristic
- development and evaluation of QA systems with increasing architectural complexity
- state-of-the-art results for FastQA and its extended version FastQAExt on three recent QA benchmarks
- in-depth discussion of usefulness of a complex interaction layer
- qualitative analysis revealing that limited functionality required by our heuristic is learnt by FastQA and thus, sufficient to achieve strong results

2 FastQA

We begin by motivating the FastQA architecture based on a heuristic that should be able to answer most questions in an extractive QA setting: i) the type of the answer span should correspond to the answer type given by the question, and ii) the correct answer should further be surrounded by a context that fits the question, i.e., more precisely by many question words. Similar heuristics were frequently implemented in traditional QA systems, e.g., in the answer extraction step of Moldovan et al. (1999).

Lample et al. (2016) demonstrate that bidirectional recurrent neural networks (BiRNN) are powerful at recognizing named entities which makes them sensible choice for context encoding to satisfy i). ii) can similarly be achieved by a RNN that is informed of the locations of question tokens appearing in the context. Finally, we argue that BiRNNs are powerful enough to model both requirements. Thus, the FastQA system has to include a BiRNN for sequentially encoding a sequence of (embedded) words with additional features that inform the encoder about the occurrence of question words.

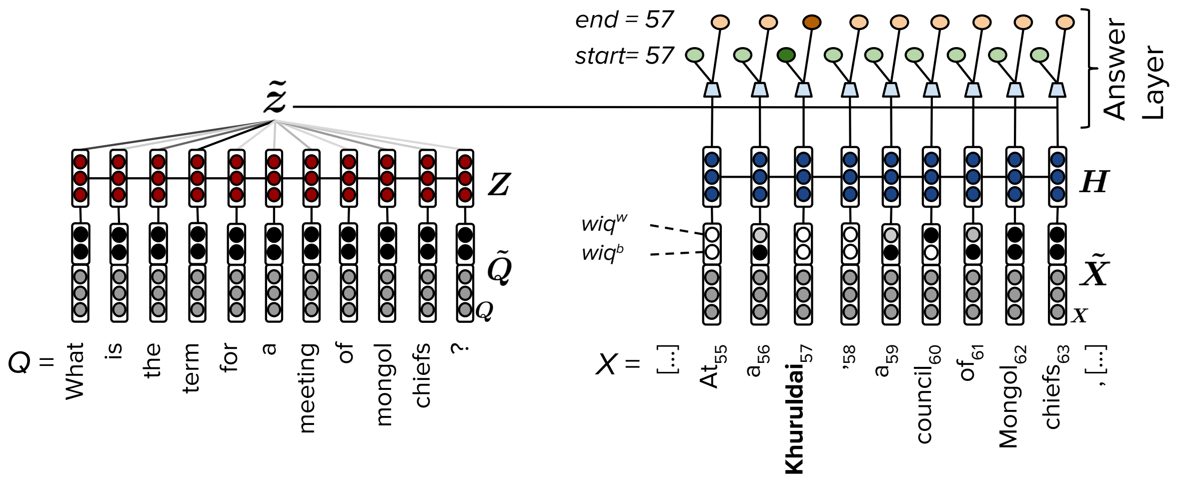
On a high level, the FastQA model consists of three basic layers, namely the embedding-, encoding and answer layer, that are described in detail in the following. We denote the hidden dimensionality of the model by n, the question tokens by $Q = (q_1, ..., q_{L_Q})$, and the context tokens by $X = (x_1, ..., x_{L_X})$. An illustration of the basic architecture is provided in Figure 1.

2.1 Embedding

The embedding layer is responsible for mapping tokens x to their corresponding high-dimensional representation $\boldsymbol{x}$. Typically this is done by mapping each word x to its corresponding word embedding $\boldsymbol{x}^w$ (*lookup-embedding*) using an embedding matrix E, s.t. $\boldsymbol{x}^w = Ex$. Another approach is to embed each word by encoding their corresponding character sequence $x^c = (c_1, ..., c_{L_X})$ with C, s.t. $\boldsymbol{x}^c = C(x_c)$ (*char-embedding*). In this work, C is a convolutional neural network with max-pooling over time as explored by Seo et al. (2017). Both approaches are combined via concatenation, s.t. the final embedding becomes $\boldsymbol{x} = [\boldsymbol{x}^w; \boldsymbol{x}^c] \in \mathbb{R}^d$.

2.2 Features

When processing the context the encoder should be aware of the words that are important for answering the question. Therefore, in addition to the embedded context words we feed two computable features as input to the encoder, namely a binary- and a weighted *word-in-question* feature which are explained in the following.



end = 57
start= 57
Answer Layer
$\tilde{z}$
Z
H
$\tilde{Q}$
Q
wiq^w
wiq^b
$\tilde{X}$
X
Q = What is the term for a meeting of mongol chiefs ?
X = [...] At$_{55}$ a$_{56}$ Khuruldai$_{57}$ '$_{58}$ a$_{59}$ council$_{60}$ of$_{61}$ Mongol$_{62}$ chiefs$_{63}$, [...]

Figure 1: Illustration of FastQA system on example question from SQuAD. The two word-in-question features (wiq^b, wiq^w) are presented with varying degrees of activation.

binary The binary word-in-question (wiq^b) feature is 1 for tokens that are part of the question and else 0. The following equation formally defines this feature where $\mathbb{I}$ denotes the indicator function:

$$\text{wiq}_j^b = \mathbb{I}(\exists i : x_j = q_i) \tag{1}$$

weighted The weighted word-in-question (wiq^w) feature combines two distinct term frequencies, a typical IR measure, namely the term frequency ($\text{tf}(x_j|Q)$) of a token x_j in the question and the inverse term frequency ($\text{itf}(x_j|C)$) in the context. The rationale behind this feature is the fact that question tokens which rarely appear in the context are more likely to be important for answering the question. We use a soft version of wiq^w to account for morphology and synonyms by exploiting the given embedded tokens. The feature is defined in Eq. 3.

$$sim_{i,j} = \boldsymbol{v}_{wiq}(\boldsymbol{x}_j \odot \boldsymbol{q}_i) \quad , \boldsymbol{v}_{wiq} \in \mathbb{R}^n \tag{2}$$
$$\text{wiq}_j^w = \sum_i \text{softmax}(sim_{i,\cdot})_j \tag{3}$$

Eq. 2 defines a basic similarity score between q_i and x_j. If we define this score to be a constant in case x_j is part of the question and $-\inf$ else, the resulting wiq^w-feature would correspond exactly to $\text{itf}(x_j|X) \cdot \text{tf}(x_j|Q)$.

Finally, to be able to use the same encoder for both question and context we fix these two features to one for the question.

2.3 Encoding

In the following, we describe the encoding of the context which is analogous to that of the question.

To allow for interaction between the two embeddings described in §2.1, they are first projected jointly to a n-dimensional representation (Eq. 4)) and further transformed by a single highway layer (Eq. 5) similar to Seo et al. (2017).

$$\boldsymbol{x}' = P\boldsymbol{x} \quad , P \in \mathbb{R}^{n \times d} \tag{4}$$
$$\boldsymbol{g}_e = \boldsymbol{\sigma}(\text{FC}(\boldsymbol{x}')) \, , \, \boldsymbol{x}'' = \tanh\left(\text{FC}\left(\boldsymbol{x}'\right)\right)$$
$$\tilde{\boldsymbol{x}} = \boldsymbol{g}_e \boldsymbol{x}' + (1 - \boldsymbol{g}_e)\boldsymbol{x}'' \tag{5}$$

For the sake of brevity use the abbreviation FC to denote a fully connected layer, i.e., $\text{FC}(\boldsymbol{x}) = W\boldsymbol{x} + \boldsymbol{b}$. Note, that these preliminary transformations are only performed when combining different kinds of embeddings, namely *char+lookup*.

The complete input $\tilde{\boldsymbol{X}} \in \mathbb{R}^{n+2 \times L_X}$ he encoder is defined as follows:

$$\tilde{\boldsymbol{X}} = ([\tilde{\boldsymbol{x}}_1; \text{wiq}_1^b; \text{wiq}_1^w], ..., [\tilde{\boldsymbol{x}}_{L_X}; \text{wiq}_{L_X}^b; \text{wiq}_{L_X}^w])$$

$\tilde{\boldsymbol{X}}$ is fed to a bidirectional RNN and its output is again projected to allow for interaction between the features accumulated in the forward and backward RNN (Eq. 6). In preliminary eriments we found LSTMs (Hochreiter and Schmidhuber, 1997) to perform best.

$$\begin{aligned} \boldsymbol{H'} &= \text{Bi-LSTM}(\tilde{\boldsymbol{X}}) &, \boldsymbol{H'} \in \mathbb{R}^{2n \times L_X} \\ \boldsymbol{H} &= \tanh(B\boldsymbol{H'}^\top) &, B \in \mathbb{R}^{n \times 2n} \end{aligned} \tag{6}$$

We initialize the projection matrix B with $[I_n; I_n]$, where I_n denotes the n-dimensional identity matrix. It follows that $\boldsymbol{H}$ is the sum of the outputs from the forward- and backward-LSTM at the beginning of training.

As mentioned before, we utilize the same encoder parameters for both question and context, except the projection matrix B which is not shared. However, they are initialized the same way, s.t. the context and question encoding is identical at the beginning of training.

2.4 Answer Layer

After encoding context X to $\boldsymbol{H} = [\boldsymbol{h}_1, ..., \boldsymbol{h}_{L_X}]$ and the question Q to $\boldsymbol{Z} = [\boldsymbol{z}_1, ..., \boldsymbol{z}_{L_Q}]$, we first compute an attention weighted, n-dimensional question representation $\tilde{\boldsymbol{z}}$ of Q (Eq. 7).

$$\begin{aligned} \alpha_i &\propto \text{[illegible]}(\boldsymbol{v}_q \boldsymbol{z}_i) &, \boldsymbol{v}_q \in \mathbb{R}^n \\ \tilde{\boldsymbol{z}} &= \sum_i \alpha_i \boldsymbol{z}_i \end{aligned} \tag{7}$$

The probability distribution p_s for the start location of the answer is computed as follows:

$$\begin{aligned} \boldsymbol{s}_j &= \max\left(0, \text{FC}\left([\boldsymbol{h}_j; \tilde{\boldsymbol{z}}; \boldsymbol{h}_j \odot \tilde{\boldsymbol{z}}]\right)\right) \\ p_s(j) &\propto \exp(\boldsymbol{v}_s \boldsymbol{s}_j) \quad , \boldsymbol{v}_s \in \mathbb{R}^n \end{aligned} \tag{8}$$

The conditional probability distribution p_e for the end location conditioned on the start location s is computed similarly as follows:

$$\begin{aligned} \boldsymbol{e}_j &= \max\left(0, \text{FC}\left([\boldsymbol{h}_j; \boldsymbol{h}_s; \tilde{\boldsymbol{z}}; \boldsymbol{h}_j \odot \tilde{\boldsymbol{z}}; \boldsymbol{h}_j \odot \boldsymbol{h}_s]\right)\right) \\ p_e(j|s) &\propto \exp(\boldsymbol{v}_e \boldsymbol{e}_j) \quad , \boldsymbol{v}_e \in \mathbb{R}^n \end{aligned} \tag{9}$$

The overall probability p of predicting an answer span (s, e) is $p(s, e) = p_s(s) \cdot p_e(e|s)$.

Beam-search To compute the answer span with the highest probability, we employ beam-search with a beam-size of k. This means that ends for the top-k answer starts are predicted during application and the span with the highest overall probability is predicted as final answer.

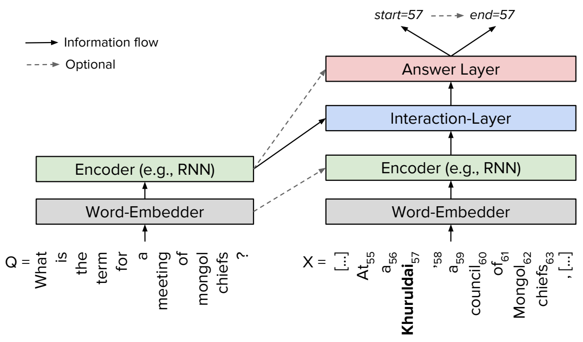

start=57
end=57
Information flow
Optional
Answer Layer
Interaction-Layer
Encoder (e.g., RNN)
Word-Embedder
Encoder (e.g., RNN)
Word-Embedder
Q = What is the term for a meeting of mongol chiefs ?
X = [...] At55 a56 Khuruldai57 '58 a59 council60 of61 Mongol62 chiefs63 , [...]

Figure 2: Illustration of the basic architecture which underlies most existing neural QA systems.

3 Comparison to Prior Work

Many neural architectures for extractive QA have been conceived very recently. Most of these systems can be broken down into four basic layers for which individual innovations were proposed. A high-level illustration of these systems is show in Figure 2. In the following, we compare our system in more detail with existing models.

Embedder The embedder is responsible for embedding a sequence of tokens into a sequence of n-dimensional states. It typically maps tokens to their pre-trained embeddings. However, as described in §2.1 these embeddings can be further combined with computed character-based word embeddings (Seo et al., 2017; Yang et al., 2017).

Encoder Embedded tokens are further encoded by some form of composition function. A prominent type of encoder is the (bi-directional) recurrent neural network (RNN). Feeding it with additional word-level features similar to ours is not novel. For instance, Wang et al. (2017) calculate word-level relevance scores for the context based on the query words that are used as multiplicative filters. We believe that such a form of gating is too strict and should rather be handled by the encoder. Li et al. (2016), on the other hand, successfully employed two simple binary features in addition to word-embeddings during encoding which is similar to our encoder.

Interaction Layer Most research focused on the interaction layer[1] which is responsible for the interaction between question and context. Different ideas were explored such as attention (Wang and

[1]Other works used different names, e.g., match layer, aggregation layer, model layer, etc.

Jiang, 2017; Yu et al., 2017), co-attention (Xiong et al., 2017), bi-directional attention flow (Seo et al., 2017), multi-perspective context matching (Wang et al., 2017) or fine-grained gating (Yang et al., 2017). All of these ideas aim at enriching the encoded context with weighted states from the question and in some cases also from the context. These are gathered individually for each context state, concatenated with it and serve as input to an additional RNN.

Answer Layer Finally, most systems divide the prediction of the answer span into predicting the start and the end using feed-forward neural networks for scoring. Their complexity ranges from using a single fully-connected layer (Seo et al., 2017; Wang et al., 2017) to employing convolutional neural networks (Trischler et al., 2017) or deep Highway-Maxout-Networks which are repeated iteratively as demonstrated by (Xiong et al., 2017). As a simpler alternative to this iterative start and end pointer prediction, we introduce beam-search in the answer layer to extract the most likely answer span. Preliminary experiments show that more complex answer layers than our 2-layer MLP did not improve performance.

4 FastQA Extended

To explore the necessity of the interaction layer, similar to previous works, we extend our model with an additional interaction layer. In particular, we employ *representation fusion* to enable the exchange of information in between passages of the context (*intra-fusion*), and between the question and the context (*inter-fusion*). Representation fusion is defined as the weighted addition between a state, i.e., its n-dimensional representation, and its respective co-representation. For each context state its corresponding co-representation is retrieved via attention from the rest of the context (intra) or the question (inter), respectively, and fused into its own representation. In contrast to recent work which typically combines these representations by concatenation and feeds them as input to an additional RNN (e.g., LSTM), our approach can be considered a more light-weight version of interaction. Because this extension is not the focus of this work and for the sake of brevity we describe technical details of this layer in Appendix A.

5 Experimental Setup

5.1 Datasets

SQuAD The Stanford Question Answering Dataset (Rajpurkar et al., 2016)[2] comprises over $100k$ questions about paragraphs of 536 Wikipedia articles. It was the first open-domain, natural language question answering dataset at that scale.

NewsQA The NewsQA dataset (Trischler et al., 2017)[3] contains $100k$ answerable questions from a total of $120k$ questions. In this work, similar to Trischler et al. (2017) we only focus on the answerable part and leave the detection of unanswerable questions for future work. The dataset is built from CNN news stories that were originally collected by Hermann et al. (2015).

MsMARCO The Microsoft Machine Reading Comprehension dataset (Nguyen et al., 2016)[4] differs from the aforementioned datasets in various aspects. It contains $100k$ real world queries from the Bing search engine and human generated answers that are based on relevant web documents. Because we focus on extractive question answering in this work, we limit the queries for training to queries whose answers are directly extractable from the given web documents. We found that 67.2% of all queries fall into this category. Evaluation, however, is performed on the entire development and test set, respectively, which makes it impossible to answer the subset of *Yes/No* questions ($\approx 7\%$) properly. For the sake of simplicity, we concatenate all given paragraphs and treat them as a single document. Since most queries in MsMARCO are lower-cased we also lower-cased the context.

5.2 Evaluation

Performance on the SQuAD and NewsQA datasets is measured in terms of *exact match* (accuracy) and a mean, per answer token-based *F1* measure which was originally proposed by Rajpurkar et al. (2016) to also account for partial matches. The official scoring measure of MsMARCO for generative models is ROUGE-L. Even though our model is extractive we use our extracted answers as if they were generated.

[2]`https://rajpurkar.github.io/SQuAD-explorer/`
[3]`https://datasets.maluuba.com/NewsQA/`
[4]`http://www.msmarco.org/`

Model	Test	
	F1	Exact
Logistic Regression[1]	51.0	40.4
Match-LSTM[2]	73.7	64.7
Dynamic Chunk Reader[3]	71.0	62.5
Fine-grained Gating[4]	73.3	62.5
Multi-Perspective Matching[5]	75.1	65.5
Dynamic Coattention Networks[6]	75.9	66.2
Bidirectional Attention Flow[7]	77.3	68.0
r-net[8]	77.9	69.5
FastQA w/ beam-size $k = 5$	77.1	68.4
FastQAExt $k = 5$	**78.9**	**70.8**

Table 1: Official SQuAD leaderboard of single-model systems on test set from 2016/12/29, the date of submitting our model. [1]Rajpurkar et al. (2016), [2]Wang and Jiang (2017), [3]Yu et al. (2017), [4]Yang et al. (2017), [5]Wang et al. (2017), [6]Xiong et al. (2017), [7]Seo et al. (2017), [8] not published.

Model	Dev		Test	
	F1	Exact	F1	Exact
Match-LSTM[1]	48.9	35.2	48.0	33.4
BARB[2]	49.6	36.1	48.3	34.1
FastQA $k = 5$	**56.4**	**43.7**	55.7	41.9
FastQAExt $k = 5$	56.1	**43.7**	**56.1**	**42.8**

Table 2: Results on the NewsQA dataset. [1]Wang and Jiang (2017) was re-implemented by [2]Trischler et al. (2017).

5.3 Training

Throughout all experiments we use a hidden dimensionality of $n = 300$, variational dropout (Gal and Ghahramani, 2015) at the input embeddings with a rate of 0.5 and 300-dimensional fixed word-embeddings from `Glove` (Pennington et al., 2014). We employed ADAM (Kingma and Ba, 2015) for optimization with an initial learning-rate of 10^{-3} which was halved whenever the $F1$ measure on the development set dropped between checkpoints. Checkpoints occurred after every 1000 mini-batches each containing 64 examples.

Because NewsQA and MsMARCO contain examples with very large contexts (up to more than 1500 tokens) we cut contexts larger than 400 tokens in order to efficiently train our models. We ensure that at least one, but at best all answers are still present in the remaining 400 tokens.

Model	Dev		Test	
	Bleu	Rouge	Bleu	Rouge
ReasoNet[1]	-	-	14.8	19.2
Match-LSTM[2]	-	-	**37.3**	**40.7**
FastQA $k = 5$	34.9	33.0	34.0	32.1
FastQAExt $k = 5$	**35.0**	**34.4**	33.9	33.7

Table 3: Results on the MsMARCO dataset (Bleu-1, Rouge-L). [1]Shen et al. (2016)- extractive model trained out-of-domain on SQuAD, [2]Wang and Jiang (2017)- method for MsMARCO not published yet.

6 Results

6.1 Empirical Results

The results presented in Tables 1, 2 and 3 clearly demonstrate the strength of the FastQA system. It is very competitive to previously established state-of-the-art results on all datasets and even improves those for NewsQA. This is quite surprising when considering the simplicity of FastQA. Our extended version FastQAExt achieves even slightly better results. It outperforms all previously reported results on the very competitive SQuAD benchmark. The strong performance of our purely extractive system on the generative MsMARCO dataset is also notable. It shows that answers to Bing queries can mostly be extracted directly from web documents without the need for a more complex generative approach.

6.2 Model Component Analysis

Model	Dev	
	F1	Exact
BiLSTM	58.2	48.7
BiLSTM + wiq^b	71.8	62.3
BiLSTM + wiq^w	73.8	64.3
BiLSTM + wiq^{b+w} (FastQA*)	74.9	65.5
FastQA* + intrafusion	76.2	67.2
FastQA* + intra + inter (FastQAExt*)	77.5	68.4
FastQA* + char-emb. (FastQA)	76.3	67.6
FastQAExt* + char-emb. (FastQAExt)	78.3	69.9
FastQA w/ beam-size 5	76.3	67.8
FastQAExt w/ beam-size 5	**78.5**	**70.3**

Table 4: SQuAD results on development set for increasingly complex architecture.

Table 4 shows the individual contributions of each model component that was incrementally added to the initial BiLSTM model. It merely consists of a fixed word-embedding layer followed by a BiLSTM encoder without additional features and our proposed answer-layer without beam search. We see that the most crucial performance boost stems from the introduction of either one of our features. However, all other extensions also achieve notable improvements between 1 and 2% with beam-search being an exception. It slightly improves results which shows that the most probable answer start is not necessarily the start of the best answer span.

In general, these results are interesting in many ways. For instance, it is surprising that a simple binary feature like wiq^b can have such a dramatic effect on the overall performance. We believe that the reason for this is the fact that an encoder without any knowledge of the actual question has to account for every possible question that might be asked, i.e., it has to keep track of the entire context around each token in its recurrent state. An informed encoder, on the other hand, can selectively keep track of question related information. It can further abstract over concrete entities to their respective types because it is rarely the case that many entities of the same type occur in the question. For example, if a person is mentioned in the question the context encoder only needs to remember that the "question-person" was mentioned but not the concrete name of the person.

Another interesting finding is the fact that additional character based embeddings have a notable effect on the overall performance which was already observed by Seo et al. (2017); Yu et al. (2017). We see further improvements when employing representation fusion to allow for more interaction. This shows that a more sophisticated interaction layer can help. However, the differences are not substantial, indicating that this extension does not offer any systematic advantage.

6.3 Do we need additional interaction?

We measured both time- and space-complexity of FastQAExt and a reimplementation of the Dynamic Coattention Network (DNC, Xiong et al. (2017)) in relation to FastQA. We found that FastQA is about twice as fast as the other two systems and requires $2-4$ times less memory compared to FastQAExt and DNC, respectively[5].

In addition, we looked for systematic advantages of FastQAExt over FastQA by comparing SQuAD examples from the development set that were answered correctly by FastQAExt and incorrectly by FastQA (589 FastQAExt wins) against FastQA wins (415). We studied the average question- and answer length as well as the question types for these two sets but could not find any systematic difference. The same observation was made when manually comparing the kind of reasoning that is needed to answer a certain question. This finding aligns with the marginal empirical improvements between the two systems indicating that FastQAExt seems to generalize slightly better but does not offer a particular, systematic advantage. Therefore, we argue that the additional complexity introduced by the interaction layer is not necessarily justified by the incremental performance improvements presented in §6.1, especially when memory or run-time constraints exist.

6.4 Qualitative Analysis

Besides our empirical evaluations this section provides a qualitative error inspection of predictions for the SQuAD development dataset. We analyse 55 errors made by the FastQA system in detail and highlight basic abilities that are missing to reach human level performance. In the following examples, predicted answers are underlined while correct answers are presented in boldface.

We found that most errors are based on a lack of either syntactic understanding or a fine-grained semantic distinction between lexemes with similar meanings. Other error types are mostly related to annotation preferences, e.g., answer is good but there is a better, more specific one, or ambiguities within the question or context.

A prominent type of mistake is a lack of fine-grained understanding of certain answer types as in the following example:

What religion did the Yuan discourage, to support Buddhism?

Buddhism (especially <u>Tibetan</u> Buddhism) flourished, although **Taoism** endured ... persecutions... from the Yuan government

Another error is the lack of co-reference resolution and context sensitive binding of abbreviations:

Kurt Debus was appointed what position for the Launch Operations Center?

Launch Operations Center (LOC) ... Kurt Debus, <u>a member of Dr. Wernher von Braun's ... team</u>. Debus was named the LOC's first **Director** .

[5]We implemented all models in TensorFlow (Abadi et al., 2015).

The model frequently fails to capture basic syntactic structure, especially with respect to nested sentences where important separators like punctuation and conjunctions are being ignored, e.g.:

On what date was the record low temperature in Fresno?

high temperature for Fresno ... set on July 8, 1905, while the official record low ... set on **January 6, 1913**

Similar to this last example, we found that the model prefers answers that are "sequentially" close to the question words rather than syntactically. This finding aligns with the context/type matching heuristic our model is based on (§2). A manual examination of errors reveals that about 35 out of 55 mistakes (64%) can directly be attributed to the plain application of the heuristic. In contrast to these negative examples we can find a lot of correct predictions as well. However, a similar analysis revealed that about 44 out of 50 (88%) analyzed positive cases are covered by our heuristic. We therefore believe that our model and, wrt. empirical results, other models as well mostly learn a simple context/type matching heuristic.

This finding is important because it reveals that an extractive QA system does not have to solve the reasoning types of Chen et al. (2016) that were used to classify SQuAD instances (Rajpurkar et al., 2016), in order to be successful. An example that showcases this issue is given in the following.

When did building activity occur on St. Kazimierz Church?

Building activity occurred in numerous noble palaces and churches ... . One of the best examples of this architecture are ... St. Kazimierz Church (**1688-1692**)

Although it seems that evidence synthesis of multiple sentences is needed to fully understand the relation between the the answer and the question entity *St. Kazimierz Church*, answering this question is easily possible by applying our heuristic. The actual answer is the only time mention close to the important question entity mention.

7 Related Work

There has been a strong research interest in building models for machine comprehension, recently. This is mostly due to the creation large scale cloze datasets such the DailyMail/CNN dataset (Hermann et al., 2015) or the Children's Book Corpus (Hill et al., 2016) which enabled the creation of end-to-end neural architectures. A thorough analysis by Chen et al. (2016), however, revealed that the DailyMail/CNN was too easy and still quite noisy, and that the adapted, simple model of Hermann et al. (2015) can achieve impressive results. New datasets were constructed to eliminate the aforementioned problems including SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2017) and MsMARCO (Nguyen et al., 2016). In contrast to cloze-style QA, these datasets contain real world questions/queries with a much greater variety of potential answers. They are therefore much better suited to evaluate machine comprehension. Nevertheless, models built for cloze-style QA datasets laid the foundation for current state-of-the-art QA models discussed in §3.

Previous question answering datasets such as MCTest (Richardson et al., 2013) and TREC-QA (Dang et al., 2007) were too small to successfully train end-to-end neural architectures and required different approaches. Traditional statistical QA systems (e.g., Ferrucci (2012)) relied on linguistic pre-processing pipelines and extensive exploitation of external resources, such as knowledge bases for feature-engineering. Other paradigms include template matching or passage retrieval (Andrenucci and Sneiders, 2005).

8 Conclusion

In this work, we introduced a simple, context/type matching heuristic for extractive question answering which serves as the basis for the development of FastQA, an efficient neural architecture for extractive question answering. FastQA is very competitive with existing, complexer neural models. We show how an extended version of FastQA outperforms most of all existing systems, achieving state-of-the-art results on three recent benchmark datasets. However, we argue that the slight performance increase does not justify the increased complexity introduced by the extension which is similarly exhibited by existing models. Our qualitative analysis reveals that our model has only little language understanding and basically learns to model our proposed heuristic. Its strong performance, however, puts results of other systems into perspective. In the future we want to extend the FastQA model to address linguistically motivated error types, e.g., fine-grained understanding and distinction of answer types or learning to respect syntax.

Acknowledgments

We thank Sebastian Riedel, Philippe Thomas and Leonhard Hennig for comments on an early draft of this work. This research was supported by the German Federal Ministry of Education and Research (BMBF) through the projects ALL SIDES (01IW14002), BBDC (01IS14013E), and Software Campus (01IS12050, sub-project GeNIE).

References

Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Rajat Monga, Sherry Moore, Derek Murray, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Oriol Vinyals, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems .

Andrea Andrenucci and Eriks Sneiders. 2005. Automated question answering: Review of the main approaches. In *ICITA*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A Thorough Examination of the CNN / Daily Mail Reading Comprehension Task. *ACL* .

Hoa Trang Dang, Diane Kelly, and Jimmy J Lin. 2007. Overview of the TREC 2007 Question Answering Track. *TREC* .

D. A. Ferrucci. 2012. Introduction to "This is Watson". *IBM Journal of Research and Development* .

Yarin Gal and Zoubin Ghahramani. 2015. Dropout as a Bayesian Approximation : Representing Model Uncertainty in Deep Learning. *ICML* .

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. *NIPS* .

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. *ICLR* .

Sepp Hochreiter and Jürgen Schmidhuber. 1997. LONG SHORT-TERM MEMORY. *Neural Computation* .

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *ICLR* .

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. *NAACL* .

Peng Li, Wei Li, Zhengyan He, Xuguang Wang, Ying Cao, Jie Zhou, and Wei Xu. 2016. Dataset and Neural Recurrent Sequence Labeling Model for Open-Domain Factoid Question Answering. *arXiv:1607.06275v1 [cs.CL]* .

Dan I. Moldovan, Sanda M. Harabagiu, Marius Pasca, Rada Mihalcea, Richard Goodrum, Roxana Girju, and Vasile Rus. 1999. Lasso: A tool for surfing the answer net. In *TREC*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms Marco: a Human Generated Machine Reading Comprehension Dataset. *NIPS* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. *EMNLP* .

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *EMNLP* .

Matthew Richardson, Christopher J C Burges, and Erin Renshaw. 2013. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text. *EMNLP* .

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. 2017. Bi-Directional Attention Flow for Machine Comprehension. In *ICLR submission*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. ReasoNet: Learning to Stop Reading in Machine Comprehension. *arXiv:1609.05284* .

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A Machine Comprehension Dataset. *ICLR submission* .

Shuohang Wang and Jing Jiang. 2017. Machine Comprehension Using Match-LSTM and Answer Pointer. In *ICLR submission*.

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2017. Multi-Perspective Context Matching for Machine Comprehension. *ICLR submission* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic Coattention Networks for Question Answering. *ICLR submission* .

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W Cohen, and Ruslan Salakhutdinov. 2017. Words or Characters? Fine-grained Gating for Reading Comprehension. *ICLR submission* .

Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. 2017. End-to-End Reading Comprehension with Dynamic Answer Chunk Ranking. In *ICLR submission*.

A Representation Fusion

A.1 Intra-Fusion

It is well known that recurrent neural networks have a limited ability to model long-term dependencies. This limitation is mainly due to the information bottleneck that is posed by the fixed size of the internal RNN state. Hence, it is hard for our proposed baseline model to answer questions that require synthesizing evidence from different text passages. Such passages are typically connected via co-referent entities or events. Consider the following example from the NewsQA dataset (Trischler et al., 2017):

Where is Brittanee Drexel from?

The mother of a 17-year-old **Rochester, New York** high school student ... says she did not give her daughter permission to go on the trip. *Brittanee* Marie *Drexel's* mom says

To correctly answer this question the representations of *Rochester, New York* should contain the information that it refers to *Brittanee Drexel*. This connection can, for example, be established through the mention of *mother* and its co-referent mention *mom*. Fusing information from the context representation $\boldsymbol{h}_{\text{mom}}$ into $\boldsymbol{h}_{\text{mother}}$ allows crucial information about the mentioning of *Brittanee Drexel* to flow close to the correct answer. We enable the model to find co-referring mentions via a normalized similarity measure β (Eq. 10). For each context state we retrieve its *co-state* using β (Eq. 11) and finally fuse the representations of each state with their respective co-state representations via a gated addition (Eq. 12). We call this procedure *associative representation fusion*.

$$\hat{\beta}_{j,k} = \mathbb{I}(j \neq k)\, \boldsymbol{v}_\beta\, (\boldsymbol{h}_j \odot \boldsymbol{h}_k)$$

$$\beta_j = \text{softmax}(\hat{\beta}_{j,\cdot}) \tag{10}$$

$$\boldsymbol{h}^{co}_j = \sum_k \beta_{j,k} \boldsymbol{h}_k \tag{11}$$

$$\begin{aligned} \boldsymbol{h}^*_j &= \text{FUSE}(\boldsymbol{h}_j, \boldsymbol{h}^{co}_j) \\ &= \boldsymbol{g}_\beta \boldsymbol{h}_j + (1 - \boldsymbol{g}_\beta)\boldsymbol{h}^{co}_j \end{aligned} \tag{12}$$

$$\boldsymbol{g}_\beta = \boldsymbol{\sigma}(\text{FC}([\boldsymbol{h}_j; \boldsymbol{h}^{co}_j]))$$

We initialize $\boldsymbol{v}_\beta$ with $\mathbf{1}$, s.t. $\hat{\beta}_{j,k}$ is initially identical to the dot-product between hidden states.

We further introduce *recurrent representation fusion* to sequentially propagate information gathered by associative fusion between neighbouring tokens, e.g., between the representation of *mother* containing additional information about *Brittanee Drexel* and those representations of *Rochester, New York*. This is achieved via a recurrent backward- (Eq. 13) followed by a recurrent forward fusion (Eq. 14)

$$\tilde{\boldsymbol{h}}^{bw}_j = \text{FUSE}(\boldsymbol{h}^*_j, \tilde{\boldsymbol{h}}^{bw}_{j+1}) \tag{13}$$

$$\tilde{\boldsymbol{h}}_j = \text{FUSE}(\tilde{\boldsymbol{h}}^{bw}_j, \tilde{\boldsymbol{h}}_{j-1}) \tag{14}$$

Note, that during representation fusion no new features are computed but simply combined with each other.

A.2 Inter-Fusion

The representation fusion between question and context states is very similar to the intra-fusion procedure. It is applied on top of the context representations after intra-fusion has been employed. Associative fusion is performed via attention weights γ (Eq. 15) between question states $\boldsymbol{z}_i$ and context states $\tilde{\boldsymbol{h}}_j$). The co-state is computed for each context state via γ (Eq. 16).

$$\hat{\gamma}_{i,j} = \boldsymbol{v}_\gamma\, (\boldsymbol{z}_i \odot \tilde{\boldsymbol{h}}_j)$$

$$\gamma_i = \text{softmax}(\hat{\gamma}_{i,\cdot}) \tag{15}$$

$$\tilde{\boldsymbol{h}}^{co}_j = \sum_i \gamma_{i,j} \boldsymbol{z}_i \tag{16}$$

Note, because the softmax normalization is applied over the all context tokens for each question word, γ_i will be close to zero for most context positions and therefore, its co-state will be close to a zero-vector. Therefore, only question related context states will receive a non-empty co-state. The rest of inter-fusion follows the same procedure as for intra-fusion and the resulting context representations serve as input to the answer layer.