In: Proceedings of the International Joint Conference on Neural Networks (IJCNN 2019), Budapest, Hungary

Question Answering with Hierarchical Attention Networks

Tayfun Alpay, Stefan Heinrich, Michael Nelskamp, Stefan Wermter
Knowledge Technology, Department of Informatics
University of Hamburg, Hamburg, Germany
{alpay, heinrich, 6nelskam, wermter}@informatik.uni-hamburg.de

***Abstract*—We investigate hierarchical attention networks for the task of question answering. For this purpose, we propose two different approaches: in the first, a document vector representation is built hierarchically from word-to-sentence level which is then used to infer the right answer. In the second, pointer sum attention is utilized to directly infer an answer from the attention values of the word and sentence representations. We evaluate our approach on the Children's Book Test, a cloze-style question answering dataset, and analyze the generated attention distributions. Our results show that, although a hierarchical approach does not offer much improvement over a shallow baseline, it does indeed offer a large performance boost when combining word and sentence attention with pointer sum attention.**

***Index Terms*—hierarchical attention networks, recurrent neural networks, pointer sum attention, question answering**

I. Introduction

Attention-based recurrent architectures have recently been successfully applied to tasks such as language modelling [1], speech recognition [2], or machine translation [3]. Recently, hierarchical attention has been proposed in the form of the Hierarchical Attention Network (HAN) and successfully been used for document classification [4]. The main idea of the HAN is to hierarchically apply attention mechanisms at the word- and sentence-level. This way, the network representations are able to mirror the hierarchical structure of documents. A different task that might highly benefit from structured representations is question answering. Assuming direct questions, e.g. about entities, relevant information is typically very sparsely distributed over documents used for question answering. That is, few sentences in the provided document relate to the asked question, and of these only a few contained words need to be queried to infer the correct answer.

Therefore, we investigate how the HAN can be used for this task and we focus on cloze-style question answering (QA). In such recent datasets for QA, queries are generated by removing a single word (e.g. a named entity) from a sentence which then has to be inferred from the remaining text. This allows one to assume that the answer to the question is a single word contained in the text document, and that the dataset provides candidate words for the answer.

The authors gratefully acknowledge partial support from the German Research Foundation DFG under project CML (TRR 169), the European Union under project SECURE (No 642667), and the NVIDIA Corporation for donating GPUs used in this project.

Although attention-based models are quite popular for QA tasks [5], [6], few hierarchical approaches with attention mechanisms have been proposed to date. Previous successful approaches for QA mostly use shallow recurrent encoders [7] that only operate on word-level or work with task-specific memory representations, such as the end-to-end memory network [8]. A hierarchical variant of memory networks has also been introduced [9] although its hierarchical memory is fixed and not learned. A hierarchical model for long text documents has been proposed by Choi et al. [10]. Different to our bottom-up approach of building hierarchical representations from word- to document-level, they present a more top-down approach where the model first selects relevant sentences before generating answers with reinforcement learning. Hierarchical models have also successfully been used for visual question answering [11] although the structure of images is very different to that of text documents.

Other work has investigated novel attention mechanisms that facilitate training specifically for QA tasks. For example, the Attention Sum Reader Network (ASRN; [5]) assumes that the answer is contained in the document, allowing it to directly point to the answer rather than inferring it from a blended representation of words as is usual in similar models.

In this paper, we investigate both approaches and propose a novel hierarchical model. We adapt the original HAN to QA tasks and extend it twofold to infer an answer from hierarchical representations: first, based on a hierarchical document vector representation (HAN-doc-vec) and second, based on pointer sum attention (HAN-ptr). We evaluate our models on the Children's Book Test (CBT; [12]) and compare them to a non-hierarchical variant to ultimately address the following research questions:

1) Does hierarchical attention facilitate the task of cloze-style question answering?
2) Does pointer sum attention work better than a blended document vector representation for integrating hierarchical representations?

II. Hierarchical Attention Network for Question Answering

In this section, we describe the suggested models. The hierarchical structure is realized by primarily adapting the Hierarchical Attention Network (HAN; [4]) to the QA task.

The main idea is as follows: at word level, our system encodes a sentence representation from word representations. At the sentence level, it learns to represent the document based on these sentence representations. This final representation is then used to find the most likely word in the document which answers the query. To realize this final step, we propose two variants. The first builds up a document vector (HAN-doc-vec) from hierarchical attention that is used to infer the correct answer from a blended representation. The second uses pointer sum attention at the final layer (HAN-ptr) in order to compute answer probabilities directly from the word and sentence attention values. Additionally, we evaluate our architectures by realizing a non-hierarchical attention model (RAN) as an appropriate baseline.

A. The Word Level

Our architecture assumes three different inputs, namely the text document D, the question q, and a list of candidate words c, which contains the correct answer. We start at the word level by embedding the document D^{inp} and the question q^{inp} into word vectors with the help of the pre-trained embedding matrices $\mathbf{E}_A$ and $\mathbf{E}_q$ (we use GloVe embeddings [13]). This gives us the embedded representations $\mathbf{D}^{emb} = \mathbf{E}_D \cdot D^{inp}$ and $\mathbf{q}^{emb} = \mathbf{E}_q \cdot q^{inp}$.

1) Word Encoding: Similar to the HAN, we use Gated Recurrent Units (GRUs; [14]) as recurrent sequence encoders throughout our architecture. Our main motivation of choosing the GRU over the Long Short-Term Memory (LSTM; [15]) is that it offers similar performance for less parameters. The GRU state $\mathbf{h}_t$ is generally computed with the help of an update gate $\mathbf{z}_t$ and a reset gate $\mathbf{r}_t$ as follows:

$$\mathbf{z}_t = \sigma(\mathbf{W}_{xz} \cdot \mathbf{x}_t + \mathbf{W}_{hz} \cdot \mathbf{h}_{t-1}), \tag{1}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_{xr} \cdot \mathbf{x}_t + \mathbf{W}_{hr} \cdot \mathbf{h}_{t-1}), \tag{2}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_{xh} \cdot \mathbf{x}_t + \mathbf{W}_{rh} \cdot (\mathbf{r}_t \otimes \mathbf{h}_{t-1})), \tag{3}$$

$$\mathbf{h}_t = (1 - z_t) \otimes \mathbf{h}_{t-1} + \mathbf{z}_t \otimes \tilde{\mathbf{h}}, \tag{4}$$

where $\tilde{\mathbf{h}}_t$ are the candidate activations, and the matrices $\mathbf{W}$ are trainable connections from the input or previous layer state to the current layer state. The σ denotes the sigmoid activation function, and $\otimes$ signifies element-wise multiplication.

We employ a bidirectional GRU (biGRU) [2], where we use one GRU layer to read the sentences forward, from left to right, and another to read them backward in reversed order. This gives us the forward hidden state $\vec{\mathbf{h}}_t$ and the backward hidden state $\overleftarrow{\mathbf{h}}_t$. Both layer encodings are then combined by concatenation, i.e. $\mathbf{h}_t = [\vec{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t]$, to receive the output of the bidirectional layer. Therefore, by looking at both directions simultaneously, we allow the network to capture more context around each word.

At the word level, we now encode our word embeddings for the document and the question with two separate biGRU layers, calling the resulting encodings $\mathbf{D}^{word}$ and $\mathbf{q}$:

$$\mathbf{D}^{word} = \text{biGRU}(\mathbf{D}_t^{emb}) = [\vec{\mathbf{h}}_t(\mathbf{D}_t^{emb}), \overleftarrow{\mathbf{h}}_t(\mathbf{D}_t^{emb})] \tag{5}$$

$$\mathbf{q} = \text{biGRU}(\mathbf{q}_t^{emb}) = [\vec{\mathbf{h}}_t(\mathbf{q}_t^{emb}), \overleftarrow{\mathbf{h}}_t(\mathbf{q}_t^{emb})] \tag{6}$$

The document is processed sentence by sentence at this word level, i.e. the currently processed sentence serves as the available context. Consequently, $\mathbf{D}_t^{word}$ encodes contextual information for each word within its sentence.

2) Word Attention: In the next step, we use an attention layer so that each word can have a different contribution to the sentence representation:

$$\beta_{i,j}^{word} = \mathbf{q}^T \cdot \mathbf{D}_{i,j}^{word} \tag{7}$$

$$\alpha_{i,j}^{word} = \text{softmax}(\beta_{i,j}^{word}) \tag{8}$$

Since the dot product acts as a similarity measure between each word and the question, $\beta_{i,j}^{word}$ acts as an indicator on the relevance of the question to the j-th word of the i-th sentence in the document encoding $\mathbf{D}_{i,j}^{word}$. Applying the softmax gives us a probability distribution which we can use as the word-level attention values $\alpha_{i,j}^{word}$.

In the final step, we use these word attention values as weights for the associated word encodings to blend them into an aggregated sentence representation:

$$\mathbf{D}_i^{blend} = \sum_{k=0}^{n-1} \alpha_{i,k}^{word} \cdot \mathbf{D}_{i,k}^{word} \tag{9}$$

This leads to words with larger attention values being represented more prominently in the sentence representations.

B. The Sentence Level

After the end of the word level, we were able to estimate the importance of each *word* w.r.t. the question. The purpose of the sentence level is to allow the same for the *sentences*. This also allows us to discard previously high word attention values in sentences with low sentence attention which are meant to contain little information to answer the question.

Therefore, the previous steps are now mostly repeated on sentence level: we feed the previously attained sentence representations into a biGRU encoder and gain a document-level representation of weighted sentences by computing attention values on the sentence representations.

1) Sentence Encoding: At this point, we can enrich the vector of each sentence with context information from the whole document by feeding the sentence representations $\mathbf{D}^{blend}$ to a sentence-level biGRU:

$$\mathbf{D}^{sent} = \text{biGRU}(\mathbf{D}_t^{blend}) = [\vec{\mathbf{h}}_t(\mathbf{D}_t^{blend}), \overleftarrow{\mathbf{h}}_t(\mathbf{D}_t^{blend})] \tag{10}$$

Since this layer processes a sequence of sentence representations, this may lead to each sentence vector $\mathbf{D}_i^{sent}$ containing information on both $\mathbf{D}_i^{blend}$ as well as the neighbouring sentences $\mathbf{D}_{i-1}^{blend}$ and $\mathbf{D}_{i+1}^{blend}$.

2) Sentence Attention: The sentence attention is computed analogous to the word attention values:

$$\beta_i^{sent} = \mathbf{q}^T \cdot \mathbf{D}_i^{sent} \tag{11}$$

$$\alpha_i^{sent} = \text{softmax}(\beta_i^{sent}) \tag{12}$$

Note that we use the same question vector q from the word level. Using the sentence encodings, β_i^{sent} now indicates the similarity between this question vector and the contextual sentence vector $\mathbf{D}_i^{sent}$.

C. Computing the Final Output

After calculating the hierarchical attention-based representations in the previous sections, we now introduce two different methods to infer an output answer. The first method uses a document vector representation (HAN-doc-vec), the second pointer sum attention (HAN-ptr) to find the correct answer word in the document.

1) Document Vector: Calculating a document vector representation after the sentence attention layer is the most natural approach as it mirrors the production of the sentence vectors $\mathbf{D}^{blend}$ on word-level. We therefore blend all weighted sentence vectors together to get $\mathbf{D}^{doc}$:

$$\mathbf{D}^{doc} = \sum_{k=0}^{m-1} \alpha_k^{sent} \cdot \mathbf{D}_k^{sent} \tag{13}$$

This vector sums up the entire document based on the sentence attention values. By doing this, we have propagated the attention information of the most-likely words from word to document level. In order to compare $\mathbf{D}^{doc}$ with the candidate answers $\mathbf{c}$, we apply a linear transformation to reduce the vector dimension to that of the embedded candidate vector $\mathbf{c}^{emb}$ and get the final document vector $\mathbf{D}^{final}$:

$$\mathbf{D}^{final} = \mathbf{W}_f \cdot \mathbf{D}^{doc} + \mathbf{b}_f, \tag{14}$$

where $\mathbf{W}_f$ is a weight matrix and $\mathbf{b}_f$ are the biases.

To compute the final output using $\mathbf{D}^{final}$ and $\mathbf{c}$, we assume that if a candidate is the answer to the question, it should be represented more prominently in $\mathbf{D}^{doc}$ than the other candidates. While it is possible to simply measure the similarity between $\mathbf{D}^{final}$ and $\mathbf{c}^{emb}$, this would lead to a negative influence of candidate vectors that do not have pre-trained embeddings. Therefore, we still initialize the candidate embeddings with pre-trained word embeddings $\mathbf{E}_c$ but allow the embedding layer to update during training (different to $\mathbf{q}^{emb}$ and $\mathbf{D}^{emb}$ who have static embeddings). Training the candidate embeddings end-to-end now allows the lower recurrent layers to enrich them with semantic context.

With the candidate embeddings, we can finally calculate the final similarity and apply a softmax to see the likelihood for each candidate c_i that it is the correct answer:

$$\gamma = \mathbf{c}^T \cdot \mathbf{D}^{final}, \tag{15}$$

$$P^{doc}(c_i|D^{inp}, q^{inp}, c^{inp}) = \text{softmax}(\gamma) \tag{16}$$

The complete HAN-doc-vec model architecture is illustrated in Fig. 1.

2) Pointer Sum Attention: Different to the HAN-doc-vec, we also investigate how we can use the previously computed attention values in order to directly infer the answer without computing a document vector after the sentence level. The main idea is inspired by the work of Kadlec et al. on their Attention Sum Reader [5] and our resulting model HAN-ptr integrates it in a hierarchical manner.

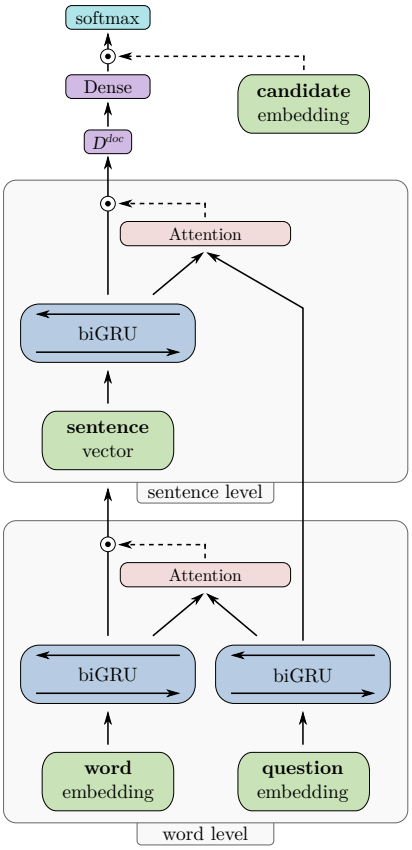


Fig. 1. HAN-doc-vec model.

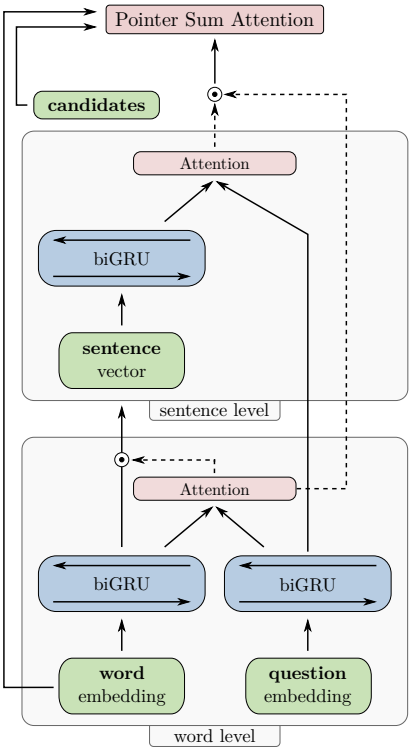


Fig. 2. HAN-ptr model.

As a first step, we combine the word and sentence attention values to get a probability distribution over all words of the document:

$$\alpha_{i,j}^{doc} = \alpha_i^{sent} \cdot \alpha_{i,j}^{word} \tag{17}$$

As a result, the overall attention $\alpha_{i,j}^{doc}$ for each word is linked with the attention it received in its sentence i as well as the attention the sentence i itself received over the document. This allows us to apply *pointer sum attention* as defined by Kadlec et al. [5] to our hierarchical model.

For the candidates c^{inp}, we now sum the attention values in $\alpha_{i,j}^{doc}$ for each occurrence of c_i^{inp} in the document D^{inp}. The candidate with the most cumulative attention is chosen as the right answer. In other words, let $Y(i)$ be the set of indices defined by:

$$Y(i) = \{(k,l) | D_{k,l}^{inp} = c_i^{inp}, 0 \leq k < m, 0 \leq l < n\}, \tag{18}$$

Then, $Y(i)$ includes all positions in the document where the candidate c_i^{inp} occurs, and the overall attention of c^{inp} is then:

$$P^{ptr}(c_i | D^{inp}, q^{inp}, c^{inp}) = \sum_{(k,l) \in Y(i)} \alpha_{k,l}^{doc} \tag{19}$$

The idea behind using pointer sum attention in such a fashion is to try to exploit the hierarchical attention structure to produce more accurate attention values than only calculating the attention on word level (as the ASRN does). While the doc-vec approach continuously transforms its representations hierarchically, this method directly uses the captured information from each hierarchical layer.

One limitation of this approach is that the pointer sum attention only looks at candidate words in the text document, i.e. any context information about other words is only coded indirectly through their attention values. The complete model using the pointer sum attention is illustrated in Fig. 2.

D. Baseline model

In order to investigate the impact of the hierarchical structure in our models, we construct a baseline by removing the sentence level from the HAN-doc-vec (illustrated in Fig. 3). The resulting model is very similar to that of Chen et al. [7] except that we still build a document vector representation from the word level, apply the linear transformation from Equation 14, and avoid using their bilinear attention form as we found this method to not increase the overall performance significantly enough to warrant a larger parameter size.

For the sake of clarity, we call this baseline the Recurrent Attention Network (RAN). One difference to the other two models is that the network takes the entire document sequence as input since the baseline only operates on word level. So different than the other two models, the recurrent layers of the RAN can keep context information between sentences.

E. Summary

To summarize, we investigate the following models with increasing complexity:

1) RAN: Recurrent Attention Network (non-hierarchical baseline)

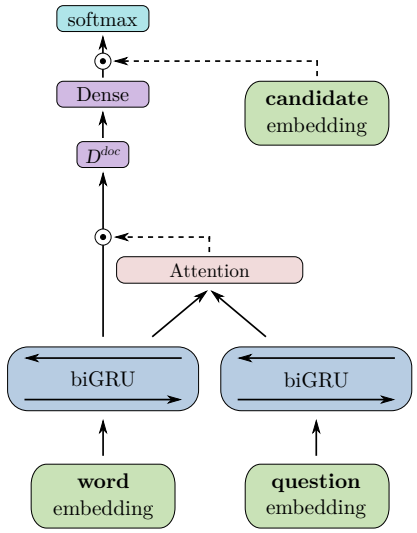


Fig. 3. The baseline model RAN.

2) HAN-doc-vec: Hierarchical Attention Network with document vector representation
3) HAN-ptr: Hierarchical Attention Network with pointer sum attention

Both HAN variants build up hierarchical representations on word and sentence level. The ptr model infers the answer directly from the attention values while the doc-vec model builds an additional representation on document level to summarize the sentences and infer the answer with trainable candidate embeddings.

III. Experiments

A. Dataset

We use the Children's Book Test (CBT; [12]) to evaluate our models. The dataset is constructed from 108 children's books which typically have a very clear narrative structure. It is also one of the currently more popular cloze-style datasets for automated question answering, wherein the questions and answers are generated by removing a single word from a sentence which then has to be inferred from the remaining text. In the CBT dataset, each document consists of 21 consecutive sentences. One word removed from the 21st sentence serves as the answer while the remaining sentence forms the query. The reader then has to read the previous 20 sentences (the context) and pick one from ten candidate answers which best fit the placeholder in the query.

For this study, we investigate two different types of words that may be treated as placeholders: Named Entities (NE) and Common Nouns (CN). These two categories have been constructed for the CBT dataset using the Stanford Core NLP toolkit [16] and are therefore effectively treated as two different datasets. We choose these categories as it has been shown that LSTM language models have difficulty achieving human performance with these two-word types in the CBT dataset [12]. It is important to note that we only use raw text as input, i.e. there are no entity annotations that are shown to the

models. CBT-CN consists of 120,769 and CBT-NE of 108,719 training documents. Both datasets have 2,000 validation and 2,500 test documents. The total number of unique words is roughly 53,000 for both datasets.

B. Training Details

We use pre-trained GloVe word embeddings [13] and investigate the three available embedding dimensions of $|\mathbf{E}| \in \{100, 200, 300\}$. For unknown words, we draw random numbers from a uniform distribution in the same value range as the embeddings. For the number of hidden units we choose $|\vec{\mathbf{h}}| = |\overleftarrow{\mathbf{h}}| \in \{256, 384, 512\}$ for both backward and forward layers, i.e. each bidirectional hidden layer has twice as many units in total. We observed smaller sizes leading to significantly worse accuracies, whereas larger layers led to challenging computational requirements. The models are run on a single GPU (NVIDIA GTX 1080 Ti) using a batch size of 64, which leads to an average training time of 20 minutes (HAN-ptr) and 25 minutes (HAN-doc-vec) per epoch on the larger dataset. The RAN model takes 2 hours per epoch since it receives the entire document as a sequence. All networks train for a maximum of 7 epochs and we record the epoch with the best validation accuracy. The recorded median number of epochs for convergence is 4 for the RAN baseline and HAN-doc-vec, and 3 for the HAN-ptr. All our models are trained using the Adam optimizer [17] and the categorical cross entropy loss. A learning rate of 0.001 is picked for both datasets as the result of preliminary experiments. Each hyperparameter configuration is trained 10 times with different seeds to account for randomization.

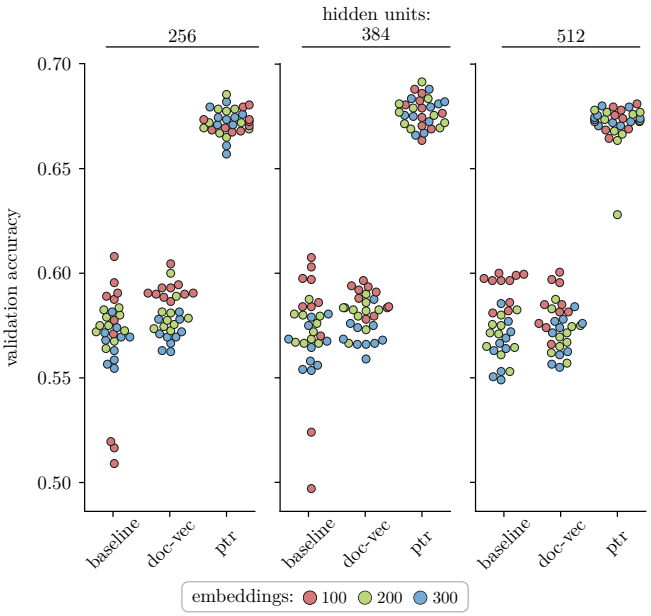
C. Evaluation

1) Hyperparameters: Fig. 4 summarizes our hyperparameter evaluation on both datasets. Overall, smaller GloVe word embedding dimensions seem to work better than larger ones. However, the variance is smaller for the HAN-ptr model, indicating that the other two models could face difficulties training candidate embeddings with high dimensionality (the HAN-ptr uses a simple lookup instead of candidate embeddings).

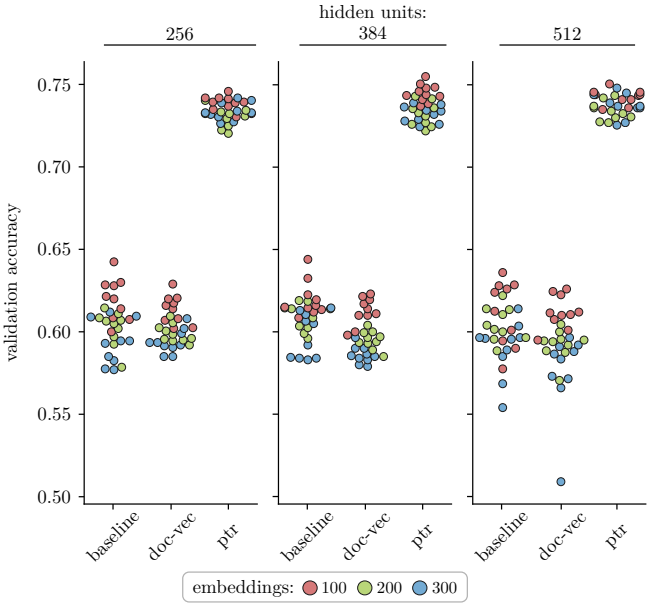
The number of hidden units per layer seems to positively correlate with the validation accuracy, although the overall variance caused by changing hidden units is much smaller than that observed from different embeddings. This shows an overall robustness of all models w.r.t. their parameter size.

As in previous studies, our networks also have an easier time querying for named entities than common nouns (see Tab. I). The HAN-doc-vec model is even slightly behind our baseline for named entities, while it gives more consistent results for common nouns where it slightly improves on the baseline.

2) Test Results: After hyperparameter optimization, we select the best models based on the validation accuracy and evaluate them on the test set. Tab. I shows our best results for the Common Noun (CN) and Named Entity (NE) categories of the CBT dataset. As can be seen, the HAN-doc-vec falls short to significantly improve on our baseline RAN (especially in the NE category). This shows that the hierarchical processing



(a) Results on CBT-CN (Common Noun)



(b) Results on CBT-NE (Named Entity)

Fig. 4. Swarmplot with validation accuracies. Evaluating the three models, RAN (baseline), HAN-doc-vec, and HAN-ptr with different embedding and hidden dimensions.

itself does not guarantee better performance. On the other hand, the HAN-ptr gives significantly better results than both the baseline and the doc-vec version. By comparing this result to the single model Attention Sum Reader Network (ASRN), we can also conclude that the attention sum pointer is not the only cause for the performance gain of the HAN-ptr. For common nouns, in particular, we get a better test accuracy with the HAN-ptr. Since the ASRN also uses the same attention mechanism, we are left to assume that our hierarchical layout is the reason that we can partially improve on the original

	Common Noun		Named Entity	
Model	valid	test	valid	test
Humans (query) [1]	-	64.4	-	52.0
Humans (query + context) [1]	-	81.6	-	81.6
Maximum Frequency (context) [1]	27.3	28.1	29.9	33.5
LSTMs (query) [1]	61.3	54.1	50.0	40.8
LSTMs (query + context) [1]	62.6	56.0	51.2	41.8
Memory Networks [1]	64.2	63.0	70.4	66.6
Attention Sum Reader Network [2]	68.8	63.4	73.8	68.6
RAN (non-hierarchical baseline)	60.8	57.4	64.4	58.7
HAN-doc-vec	60.4	56.4	62.9	57.7
HAN-ptr	69.1	67.7	75.5	69.9

TABLE I
OVERVIEW AND COMPARISON OF OUR RESULTS ON THE CBT DATASET FOR THE COMMON NOUN (CN) AND NAMED ENTITY (NE) CATEGORIES. RESULTS MARKED WITH [1] ARE FROM [12], WITH [2] ARE FROM [5].

ASRN. The differences between the two word type categories additionally suggest that a hierarchical pointer sum approach is more beneficial when learning common nouns.

However, even though our result improves on related approaches, it does not achieve state-of-the-art results on the CBT dataset. Using a gated attention mechanism, Yang et al. [18] have achieved higher test accuracies of 72.0% (CBT-CN) and 74.9% (CBT-NE) although their model works on character- and word-level and is therefore assumed to be significantly more computationally expensive.

D. Pointer Sum Attention on Pre-Trained Models

The pointer sum attention mechanism leads to an attention distribution over the entire document, leading to more sparsely distributed attention than in the HAN-doc-vec. This means that the pointer sum attention should identify less crucial words. To gain a deeper understanding of the differences between HAN-doc-vec and HAN-ptr, we apply pointer sum attention at test time, exchanging the output layers of the best pre-trained HAN-doc-vec and RAN model with the ptr layer. This results in the baseline accuracy dropping from 55.28% to 41.08% and the HAN-ptr dropping slightly from 56.24 to 53.64%.

Since both networks have trained with a different objective, it is somewhat predictable that this procedure reduces the overall performance. It is however interesting to note, that the HAN-doc-vec achieves almost the same performance when removing everything after the final sentence attention layer and inferring the answer directly with the two attention layers. Our interpretation of this is that during training, the HAN-doc-vec only transforms the information already contained in the attention values into the document vector. It does not seem to gain much additional information by combining the contextual sentence vectors and attention values to form A^{doc}. It seems that the main challenge for the model is to come to the point of calculating the attention values, which seems to be most critical for the final performance.

We, therefore, hypothesize that HAN-doc-vec puts enough information into the attention distribution (without losing a significant amount of accuracy) to be able to evaluate it and to transform the attention into an answer while neglecting the associated contextual sentence vectors. HAN-ptr is trained to do exactly this and should, therefore, have an advantage, which may result in the better performance that we have observed.

	Common Noun			Named Entity		
	baseline	doc-vec	ptr	baseline	doc-vec	ptr
1.	business	business	am	the	the	.
2.	obey	pay	obey	and	,	,
3.	am	obey	business	in	and	the
4.	so	in	so	a	to	”
5.	in	meddle	soon	of	he	!

TABLE II
WORDS WITH THE HIGHEST **CUMULATIVE** WORD ATTENTION SCORE IN THE TEST SET (IN DESCENDING ORDER).

	Common Noun			Named Entity		
	baseline	doc-vec	ptr	baseline	doc-vec	ptr
1.	crouched	slanderous	blarney	separate	july	somalo
2.	wails	murderous	jewelled	humph	ein	ralston
3.	agreement	seaweed	shirtsleeves	sixteen	ralston	bandmaster
4.	orders	disappear	somalo	gulped	saxby	shelmardine
5.	scraps	resolute	rescue	modern	dancer	emperor

TABLE III
WORDS WITH THE HIGHEST **RELATIVE** WORD ATTENTION SCORE IN THE TEST SET (IN DESCENDING ORDER).

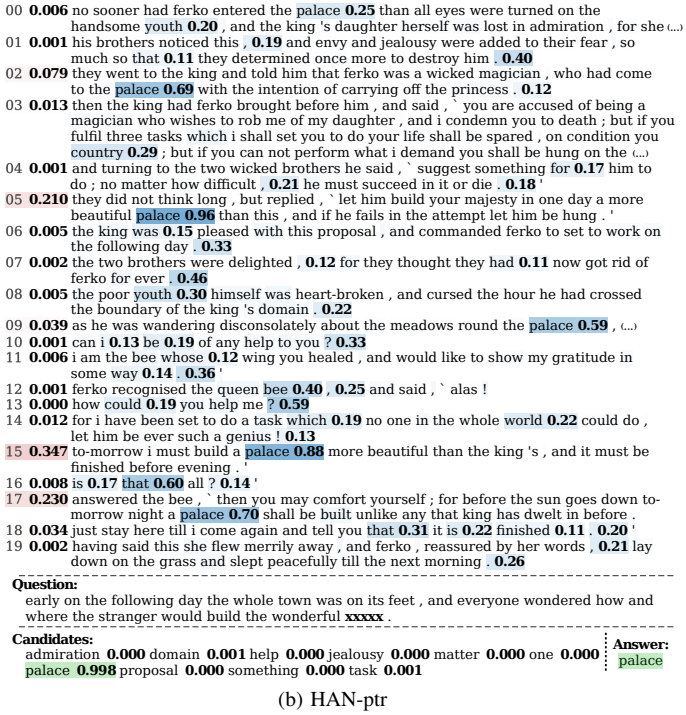
E. Attention Distributions

We investigate our models for differences in their attention distribution. For this purpose, we sort words by their total cumulative attention score over the test set. The “most attended” words are shown in Tab. II. Within each category, the models mostly share similar words. However, the differences between the two word categories are quite large. In particular, when querying named entities, a lot of attention is put on stop words. The HAN-ptr seems to even generate large attention towards punctuation.

In order to filter out very frequent words, we normalize the total attention score by word frequency which gives us a different list (see Tab. III). It is visible that these words, with the highest attention score per occurrence, are very context-specific nouns and adjectives, with many of them being uncommon. By comparing this to the absolute attention scores, we can infer that attention is often distributed to meaningless words. However, in relative terms, the attention scores are highest when rare words are encountered that share a semantic context with the answer. Since we can further see that all three models produce quite different word lists when sorted by relative attention scores, this also indicates that all three approaches reach quite different internal models for semantic context.

Since the attention distributions naturally depend on the length of the documents, we have further examined how the testing accuracy depends on the length of the context

(a) HAN-doc-vec



(b) HAN-ptr

Fig. 5. Illustration of an example document and the calculated word (blue) and sentence (red) attention values. All attention values above 0.1 have been inserted and marked in the text document. For the HAN-ptr, the overall attention of each word is its word attention multiplied with the sentence attention. The right answer is "palace".

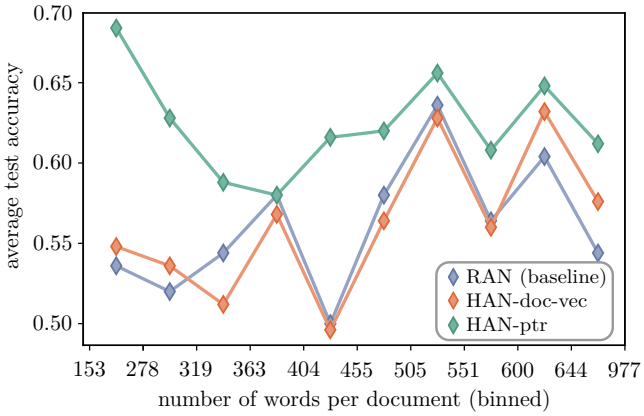
document as shown in Fig. 6. We found that the RAN and the HAN-doc-vec show particular strengths for longer documents on common nouns and shorter documents on named entities, while the HAN-ptr provides the best results independent of the document lengths. From detailed data analysis we have indications that these differences are due to the focus of RAN and HAN-doc-vec on specific words and sentences in the documents, which were particularly informative for longer documents on common nouns and for shorter documents on named entities, while HAN-ptr is a bit more independent of the document structure.

F. Visualizing the Attention

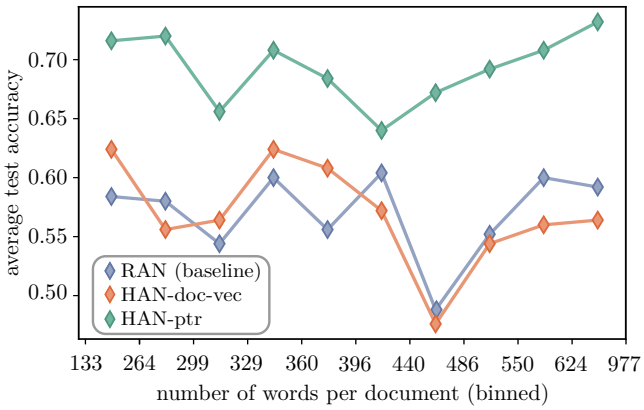
In addition to our previous analysis on the attention distributions, we inspected the attention values in detail. Two such examples can be seen in Fig. 5. In the shown document, both the HAN-doc-vec and the HAN-ptr manage to correctly identify the right answer. The HAN-ptr, however, focuses on punctuation and stop words noticeably often (see also Sec. III-E). Although this observation seems counter-intuitive, this model, interestingly, is able to create large attention on the relevant sentences in which the answer word is contained. Other candidates such as "task" (e.g. sentence 14 in Figure 5b) receive little to no attention even though they are prominently displayed in the text. This indicates that the generation of mostly "useless" attention values, does not seem to have a major influence on the accuracy of the networks. For the HAN-ptr, this can be explained by its final step in which all attention values are discarded, except for those of the candidates. This could mean that the HAN-ptr explicitly focuses on stop words and punctuation in the absence of embeddings which are semantically related to the candidates, as this would never decrease the loss.

For the HAN-doc-vec and RAN approach, frequent words that do not share any context with the candidates, are also effectively discarded due to the final dot product which measures the similarity between document representation and the question. In this model, "noisy" attention on frequent words can also be observed (e.g. "the" and "to" in Figure 5a). However, the trainable candidate embeddings seem to play an important role as the network is very certain of the right answer "palace", even though the word itself receives little attention in the entire document. Note that the attention for the 15th sentence is very high. This means that, even though on word-level "palace" receives little attention, the network was able to localize the correct answer with high certainty from its surrounding context and encode this into its sentence representation.

However, the examples also show that the sentence attention mostly serves to identify the location of the answer on sentence level. Different to the word level with its embedded context vectors, the sentence level encodings do not seem to relate or complement each other. This may be one of the reasons why the hierarchical models improve the performance only to a limited extent.



(a) Results on CBT-CN (Common Noun)



(b) Results on CBT-NE (Named Entity)

Fig. 6. Average test accuracy for different bins of document lengths.

IV. Conclusion

In this paper, we have evaluated how Hierarchical Attention Networks can be used for cloze-style question answering tasks. We have presented two approaches for building hierarchical representations based on attention mechanisms. Our results indicate that a hierarchical structure itself does not always necessarily lead to better training but that it depends on how the information from the hierarchical representations is aggregated at the output layer. The pointer sum method has shown itself to harmonize well with the hierarchical architecture in our results. Notably, our results show that recurrent hierarchical models for complex tasks can be designed with a comparably low parameter count and relatively fast training times.

When examining the sentence attention, it can be seen that the attention values themselves mostly correlate with the likelihood of containing the correct answer. This seems to help training, especially for the ptr model which weights word attention with this likelihood.

One limitation of our approach is that we assume the output answer to be contained in the document. On the other hand, the pointer sum attention is specifically designed for this setup [5]. Additionally, we assume that the network can be provided with candidates, i.e. in the introduced form, our models are able to be trained with cloze-style datasets but would require external knowledge or the generation of candidates for more open questions. Future work should therefore concentrate on evaluating our approach on different types of QA datasets. For example, one could continuously sample from the output layer in order to generate answer sentences from the pointer sum attention model. Related work has also shown that gated attention mechanisms and character-level representations can drastically improve task performance [18]. Therefore, we plan to introduce a character level to the Hierarchical Attention Network in the future.

References

[1] K. Tran, A. Bisazza, and C. Monz, "Recurrent memory networks for language modeling," in *Proceedings of the NAACL-HLT*, 2016, pp. 321–331.

[2] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4945–4949.

[3] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[4] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 NAACL-HLT*, 2016, pp. 1480–1489.

[5] R. Kadlec, M. Schmid, O. Bajgar, and J. Kleindienst, "Text understanding with the attention sum reader network," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, vol. 1, 2016, pp. 908–918.

[6] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 28, 2015, pp. 1693–1701.

[7] D. Chen, J. Bolton, and C. D. Manning, "A thorough examination of the cnn/daily mail reading comprehension task," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, vol. 1, 2016, pp. 2358–2367.

[8] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," in *Advances in neural information processing systems (NIPS)*, vol. 28, 2015, pp. 2440–2448.

[9] S. Chandar, S. Ahn, H. Larochelle, P. Vincent, G. Tesauro, and Y. Bengio, "Hierarchical memory networks," *arXiv preprint arXiv:1605.07427*, 2016.

[10] E. Choi, D. Hewlett, A. Lacoste, I. Polosukhin, J. Uszkoreit, and J. Berant, "Hierarchical question answering for long documents," *arXiv preprint arXiv:1611.01839*, 2016.

[11] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, vol. 29, pp. 289–297.

[12] F. Hill, A. Bordes, S. Chopra, and J. Weston, "The goldilocks principle: Reading children's books with explicit memory representations," *arXiv preprint arXiv:1511.02301*, 2015.

[13] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[14] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[15] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[16] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] Z. Yang, B. Dhingra, Y. Yuan, J. Hu, W. W. Cohen, and R. Salakhutdinov, "Words or characters? fine-grained gating for reading comprehension," *arXiv preprint arXiv:1611.01724*, 2016.