Information Retrieval 2
Joint Knowledge Selection and Dialogue Response Generation

Richard Olij
University of Amsterdam
olij.richard@gmail.com

Ruben Gerritse
University of Amsterdam
rgerritse95@gmail.com

Matthew van Rijn
University of Amsterdam
matthew.van.rijn@gmail.com

Melissa Tjhia
University of Amsterdam
melissa@tjhia.com

ABSTRACT

An end-to-end model that uses explicitly linked background knowledge to generate human-like responses in a dialog system setting is proposed. The background knowledge is used as templates and the model weights these by using a bilinear layer. This results in a new template from which a response is generated. To boost training this layer, an additional loss has been incorporated. However, we failed to balance the extra loss correctly, resulting in responses that could be off-topic because wrong background knowledge is often used. Even though the model generates convincing sentences, it does not outperform the BiDAF baseline.

ACM Reference Format:
Richard Olij, Ruben Gerritse, Matthew van Rijn, and Melissa Tjhia. 2019. Information Retrieval 2 Joint Knowledge Selection and Dialogue Response Generation. In *Proceedings of ACM Woodstock conference (SIGIR'2019).* ACM, New York, NY, USA, Article 4, 7 pages.

1 INTRODUCTION

One of the primary objectives of Artificial Intelligence has been the construction of dialog systems capable of conversing in a natural and coherent manner with a human. Dialog systems are being utilised in a broad range of applications such as customer service, counselling and scheduling.

Various research has already been conducted in the domain of dialog systems, however, most research has been done without incorporating the notion of background knowledge (Vinyals and Le [10], Li et al. [4], Serban et al. [9]). Their models view conversations as a sequence-to-sequence generation task, which is not the case for humans as they rely on this background knowledge for conversations. These models often produce syntactically incorrect and incoherent (off-topic) responses.

In order to produce models which output more coherent responses, various works have tried to integrate external knowledge sources from large-scale datasets (Lowe et al. [5], Serban et al. [8], Guss et al. [3], Ghazvininejad et al. [2]). However, as the external background knowledge was not explicitly linked to the utterances in the dataset, the quality of the background knowledge was not guaranteed.

Moghe et al. [6] constructed a dataset where the utterances are explicitly linked to external background knowledge with the goal to further facilitate the development of background aware conversation models. Inspired by Cao et al. [1], we propose a model which uses this explicitly linked background knowledge as soft templates. The informativeness of the templates are measured using a bilinear model and are used to create a single weighted template, which in turn is used by the decoder, together with the input sentence, to generate a response. In contrast to Cao et al. [1], our bilinear model receives feedback from the loss of the decoder during the training phrase, which in terms should boost the performance of the bilinear model.

The contributions of this paper are summarised as follows:

- We use explicitly linked background knowledge as soft templates in order to make the generated responses more coherent.
- We weight the candidate templates based on informativeness which allows backpropagation from the decoder loss through the bilinear model.

The code and generated sentences are publicly available at https://github.com/meltjh/ir2_chatbot.

2 RELATED WORK

There already has been active research conducted in the domain of training dialog systems. Vinyals and Le [10] used a simple language model based on the seq2seq framework to train a dialog system. Li et al. [4] proposed to train a dialog system using persona-based models for handling the issue of speaker consistency. Serban et al. [9] trained a dialog system using a hierarchical recurrent neural network generative model. All these models could be considered as end-to-end conversation systems which treat dialog as a sequence generation task. These models often output very short and generic answers or were inconsistent in answering semantically similar questions.

In order to improve the coherency of dialog systems, research has been conducted in the integration of external background knowledge into dialog systems. This is due to the fact that humans also rely on background knowledge during conversations. Many works use large-scale datasets as a source for external knowledge. Lowe et al. [5] proposed to incorporate the unstructured textual knowledge source Ubuntu manpages into a neural dialogue system trained

SIGIR'2019, July 2019, Paris, France

ACM ISBN .

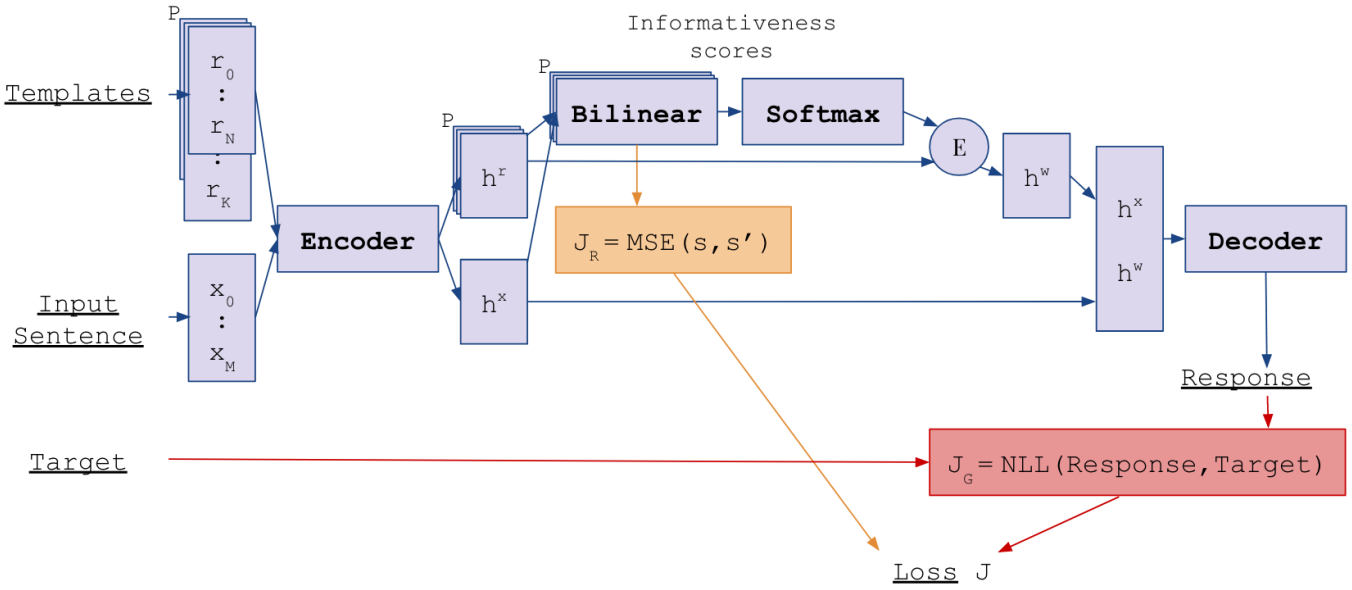


Figure 1: The proposed model

on the Ubuntu Dialogue Corpus. Serban et al. [8] presented a large-scale ensemble-based dialogue system framework, which used Reddit pages, Amazon's Evi Service, and large databases like OMDB, Google Knowledge Graph and Wikidata as sources for external knowledge. The Eigen system proposed in Guss et al. [3] selects a source for external knowledge out of a variety of conversational datasets and English language datasets. Ghazvininejad et al. [2] used Foursquare tips as external knowledge to a seq2seq framework for social media conversations.

Large-scale datasets are extract from online forums which are inherently noisy, and therefore the datasets these works use are noisy as well. In contrast to these works, Moghe et al. [6] used crowdsourcing to construct their dataset. The people who where involved in the construction of their dataset, were explicitly instructed to use only clean sentences from the external knowledge sources, thus reducing the level of noise in the dataset.

3 MODEL

The model we propose uses an encoder-decoder architecture to map an input sequence together with a variable number of template sequences to a target sequence. In the dialog system setting, the input sequence is the question or comment submitted by the user. The template sequences are a collection of sentences of background knowledge about the topic of conversation. Each sentence in the background knowledge is considered a single template. The target sequence is the response given by the system to the user's input.

The model consists of three main components. The encoder (Section 3.1) encodes the input and all templates separately. The template weighting component (Section 3.2) predicts the informativeness of each template and uses this as a weight to create a single representation of all templates, which is combined with the input encoding. Finally, the decoder (Section 3.3) uses this to generate a response to the input. A visual overview of the model is shown in Figure 1.

3.1 Encoder

The encoder receives the user's query, $\mathbf{x}$, and a variable amount P of templates, $\mathbf{r}^{(\mathbf{0})}, ..., \mathbf{r}^{(\mathbf{P})}$, as input. The number of templates $\mathbf{P}$ is different for each question and is dependent on the amount of background knowledge available for that question. The inputs are represented as embedded sequences of varying lengths. Each input is passed through the same two-layer bidirectional LSTM, to obtain the hidden states for each word. For example, the hidden state of word x_i in the question is given by:

$$\overrightarrow{\mathbf{h}}_i^x = LSTM(x_i, \overrightarrow{\mathbf{h}}_{i-1}^x)$$

Similarly, there is also the backwards hidden state $\overleftarrow{\mathbf{h}}_i^x$. The encoding of the user query is obtained by concatenating the backwards hidden state of the first word, $\overleftarrow{\mathbf{h}}_0^x$, with the forward hidden state of the final word, $\overrightarrow{\mathbf{h}}_{-1}^x$, resulting in:

$$\mathbf{h}^x = [\overleftarrow{\mathbf{h}}_0^x; \overrightarrow{\mathbf{h}}_{-1}^x]$$

The same procedure is used to obtain the encodings of the templates $\mathbf{h}^{r_i}$. The size of the LSTM's hidden state varies between experiments, but since the encodings consists of two concatenated hidden states their size is twice as large.

3.2 Weighting templates

After obtaining the individual encodings of the query and templates, the next step is to combine the external knowledge from the templates with the query into a single encoding, which can be used as input to the decoder. This encoding is created by concatenating an encoding of all templates with the query's encoding.

Not all templates are equally informative. For example, if a user asks a question about a character, a template containing information about characters is more informative than a template about the film's setting. Therefore, the single encoding of all templates is formed by taking a weighted mean of the individual templates.

The weights of the templates are determined by their predicted informativeness with respect to the question, as predicted by a bilinear layer. The informativeness is obtained by taking the sum of the ROUGE-1 and ROUGE-2 scores of the input and the target. The ROUGE scores are calculated by the ROUGE implementation of Moghe et al. [6], which differ from the PythonROUGE implementation[1], as explained in Section 5.5. The bilinear layer takes as input the query encoding $\mathbf{h}^x$ and the template encodings $\mathbf{h}^{r^{(0)}}, ..., \mathbf{h}^{r^{(P)}}$ and outputs an informativeness score for each input-template pair:

$$s(\mathbf{x}, \mathbf{r}^{(i)}) = Bilinear(\mathbf{h}^x, \mathbf{h}^{r^{(i)}})$$

The weight of each template is determined by applying the softmax function to the informativeness scores:

$$w^{(i)} = \frac{e^{s(\mathbf{x}, \mathbf{r}^{(i)})}}{\sum_{j=1}^{P} e^{s(\mathbf{x}, \mathbf{r}^{(j)})}}$$

Using these weights the encodings of the templates are combined into a relevance-weighted encoding of all templates $\mathbf{h}^r$.

$$\mathbf{h}^w = \frac{1}{P} \sum_{j=1}^{P} w^{(j)} \mathbf{h}^{r^{(j)}}$$

By concatenating this onto the query encoding the combined encoding $\mathbf{h}$ is obtained, which can be used by the decoder to generate a response to the question.

$$\mathbf{h} = [\mathbf{h}^x; \mathbf{h}^w]$$

In addition to being trained by the decoder loss, the bilinear layer is also trained directly to predict informativeness using precomputed scores. This helps the model select informative templates with less training. For each template, the informativeness loss J_R is computed as the mean squared error between the predicted and precomputed scores.

$$J_R = \sum_{j=1}^{P} (s^j - s'^j)^2$$

In this formula, s is the predicted informativeness, s' the precomputed informativeness and P the number of templates linked to the input.

3.3 Decoder

Using the combined query-template encoding $\mathbf{h}$, the decoder can generate a sentence. In this model the generated sentence represents the model's response to the input query. The decoder consists of a single GRU, where $\mathbf{h}$ is used as the initial hidden state. Since $\mathbf{h}$ is the concatenation of two encoder outputs with two times the encoder's LSTM hidden size, the GRU must have a hidden size four times the encoder's LSTM hidden size.

[1] github.com/tagucci/pythonrouge

The decoder is used in different ways during training and testing. During training, the GRU receives the entire target sequence and predicts the next word for each target word. The objective is to minimise the difference between the generated output and the target answer. Since one word is predicted for each word in the target, the output has the same length. The difference can therefore be obtained by calculating the Negative Log-Likelihood Loss between them:

$$J_G = \frac{1}{N} \sum_{i=1}^{N} -x_{i,t_i}$$

where N is the length of the target sentence t, x_{i,t_i} is the probability predicted by the decoder corresponding to the target word t_i.

During testing, the decoder must generate an entire sentence instead of just predicting the next word. This is achieved by passing the beginning of sentence tag (<bos>) to the decoder and using the generated word as the next input. An entire response is generated by repeating this step either until the end of sentence tag (<eos>) is predicted, or a predefined maximum length is reached. This maximum length has been set to 50.

3.4 Training

The final loss that is being optimised is the weighted sum of the two separate losses

$$J = \alpha J_R + (1 - \alpha) J_G$$

where α is used to weight the individual loss functions. The loss is minimised by the Adam optimiser with a learning rate of 10^{-3}.

4 EXPERIMENTS

The experiments are done using two versions of the dataset and the use of with and without external knowledge is compared as well as the vocabulary size.

4.1 Dataset

The dataset created by Moghe et al. [6] consists of 9,071 chats about 921 popular movies. Their plot, reviews, comments and fact table are considered the background knowledge and are used as the templates in the model. The chats consists of questions and answers, and are split such that one datapoint will only have one question and answer. This gives 43,192 datapoints in total, which is split into the training, validation and test sets as 80%, 10% and 10% respectively. A movie can only occur in one of the sets to ensure there is no overlap in data.

For our model, we used the same datasets as Moghe et al. [6] used for their best performing baseline model, BiDAF:

- Oracle-short: only the resource from which the answer comes from is used as the background knowledge. For example, if the target comes from the plot of the movie, only the plot is used as background knowledge and the review and comments are discarded.
- Mixed-short: a combination of the plot, review and comments is used as the background knowledge, proportional to their lengths.

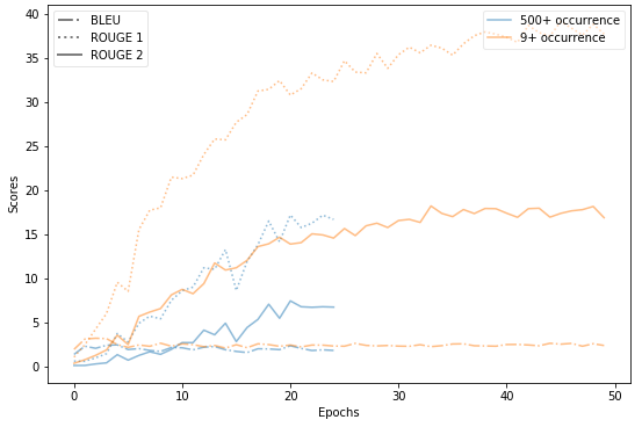


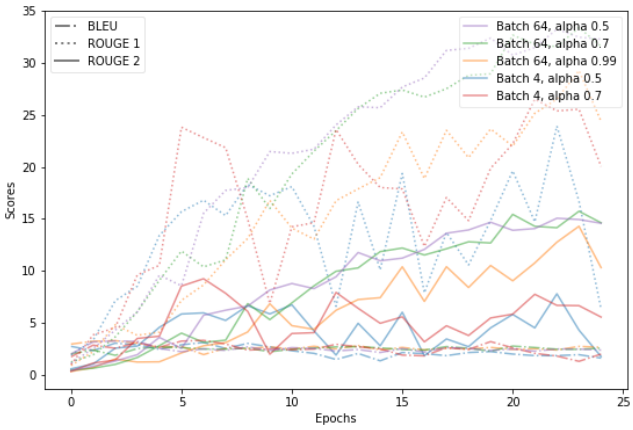
Figure 2: Comparing the minimal occurrence threshold of 500+ and 9+.

The maximum total length for background knowledge for both datasets is 256 words. According to Moghe et al. [6], their model worked best on the mixed-short dataset. Their hypothesis for this is that their model learned to recognize the sentences from the other resources as noise, and to focus on the sentences that were from the real resource. However, this was not verified.

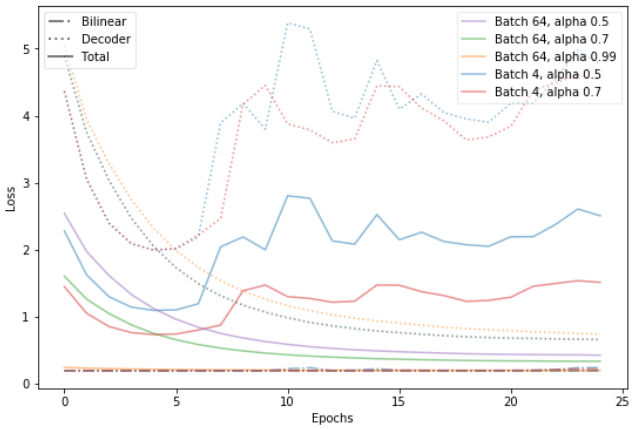
We used the pretrained 100 dimensional GloVe embeddings which were trained on Wikipedia 2014 and Gigaword 5 (Pennington et al. [7]). Similar to Moghe et al. [6], all the chat data was lowercased and tokenized with the nltk package. The vocabulary consists of the words of which the frequencies exceed a predefined threshold and for which a GloVe embedding is present. We performed our experiments with two different thresholds. The first one is similar to what Moghe et al. [6] used for their BiDAF baseline, namely only using the words that occur more than 500 times. This gives a vocabulary of 1,512 words. Since this is most likely a too little vocabulary which could not generate correct and expressive sentences, it will be compared to a larger vocabulary. This one is taking the words that occur more than 9 times, which gives 20,031 words, to get a vocabulary size similar to their HREF model. It is expected to perform better and generate more expressive and varying sentences. All the words that occur less frequent or equal to the threshold are mapped to an <UNK> token.

4.2 Knowledge

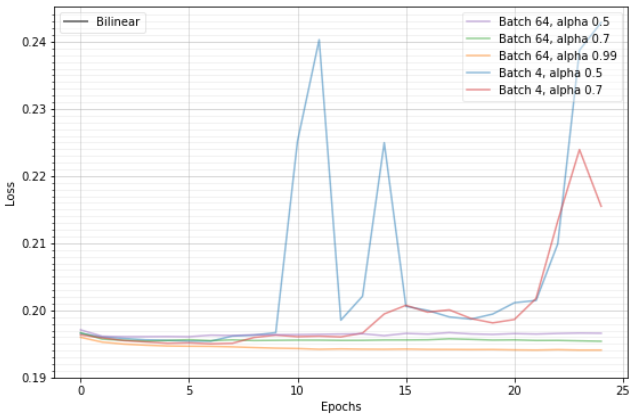
To test whether the use of templates and thus background knowledge actually improved the model, we also performed experiments without the use of templates. This is done by removing the part in which the informativeness scores of the templates are involved, namely the bilinear layer. The input sentence $\mathbf{x}$ is encoded to obtain its representation $\mathbf{h}^x$. By using this as input to the decoder, the response is generated. This results in a simple encoder-decoder model which should perform significantly lower if adding the bilinear layer to encode the background knowledge does work.



(a) Scores



(b) Losses



(c) Bilinear losses only

Figure 3: Comparing alpha values of 0.5, 0.7, 0.99 for batch size of 64 and alpha 0.7 for a batch size of 4.

5 RESULTS

The results in this section are based on default parameters unless explicitly stated otherwise. These are batch size of 64, hidden dimension of 256 (note that the bilinear has 2×256 in that case), word embedding of 100, a minimal occurrence of 9+ for the oracle-short dataset and alpha of 0.5. As mentioned in the next paragraph, training the model with the default settings longer than 25 epochs results in an increasing loss and only an insignificant increase in the ROUGE 1, ROUGE 2 and BLEU scores. Therefore, other experiments do not run for more than 25 epochs. For the other models it turned out that after 25 epochs the differences were already obvious, even when they were not fully converged yet.

5.1 Vocabulary size

Figure 2 shows the performances of using a vocabulary containing the words occurring at least 500 times and a vocabulary that contained the words occurring at least 9 times. The figure demonstrates that the small vocabulary of only 500+ occurring words performs significantly worse than 9+. This is clearly the case after the 25th epoch, and learning the default model more does not increase its performance significantly. The reason why the model with 500+ occurrences performs worse is most likely because of the small vocabulary where a great amount of the words are mapped to <UNK>. As many words are mapped to <UNK>, the input and templates also contain more <UNK>, meaning that the decoder has much less context information to work with as <UNK> has no meaning, resulting in more off-topic generated responses. Furthermore, the generated sentences also consist of a lot of <UNK> and since the targets do not contain <UNK> this results in a low performance on the evaluation metrics.

5.2 α-tuning and batch size

Figure 3 shows the results of reducing the batch size and of changing the parameter α. Decreasing the batch size does result in a significantly worse performance and learning curve, as decreasing the batch size causes the decoder loss to fluctuate. This may be due to the fact that a smaller batch size results in a higher variance of the gradients. A possible solution may be decreasing the learning rate, however, this makes the training much slower. As the training already took too long, we could not afford to test this hypothesis.

The performance of both an α of 0.5 and 0.7 perform equally in terms of the ROUGE scores. Increasing it to 0.99 performs slightly worse. The reason seems clear when looking at the losses in Figure 3b. Whereas the losses do not change that much between the different α values, the bilinear loss never seem to decrease significantly, indicating that either the bilinear layer does not learn or that the bilinear loss does not fit the problem. However, Figure 3c demonstrates that the loss does slowly and steadily decrease. Furthermore, it shows that an increase in α yields an increasing learning speed. This indicates that the problem lies in balancing the losses, rather than the loss itself.

5.3 Effect of background knowledge

Figure 4 shows the performance of the model with and without the inclusion of the bilinear layer. The figure demonstrates that the performance of the model without the bilinear layer does not increase over time[2], whereas the bilinear model shows a clear learning curve. This implies that the bilinear layer, thus including background knowledge, is necessary to predict the target sentences for this data and therefore the loss function is probably not optimal.

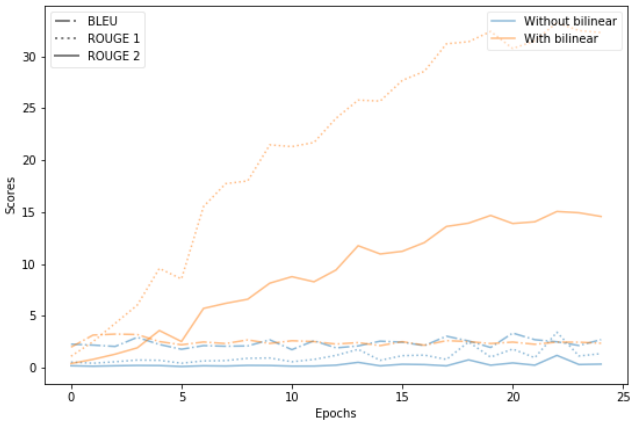


Figure 4: Comparing the proposed model with a bilinear layer for external knowledge with an encoder-decoder architecture without knowledge.

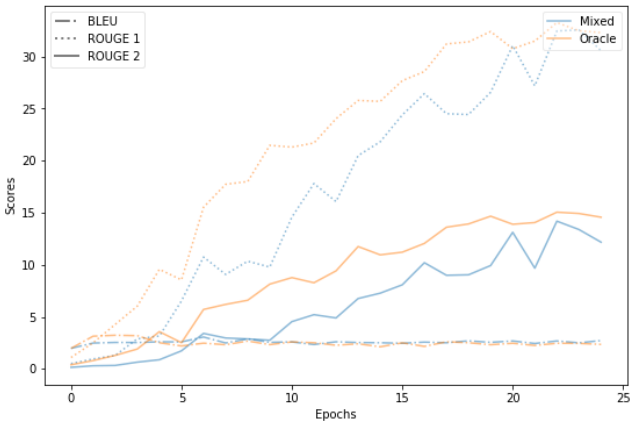


Figure 5: Comparing the oracle-short dataset with mixed-short dataset.

5.4 Oracle-short vs mixed-short

Figure 5 demonstrates the results of both the oracle-short and the mixed-short datasets. Even though mixed-short was expected to perform better due to hypothesis that the model could learn to filter out noise similarly to Moghe et al. [6], it does performs slightly worse. This could be because our model creates a new template based on the weighted sum of all templates, including the noisy ones.

[2]Note that the model does learn, and its loss does decrease similarly to the bilinear version of the model, but the scores do not increase as steadily.

Model	Dataset	BLEU	ROUGE 1	ROUGE 2
BiDAF	Oracle	34.94	46.49	40.58
BiDAF	Mixed	39.39	50.53	45.02
Proposed	Oracle	2.35	32.30	14.57
Proposed	Mixed	2.72	30.57	12.17

Table 1: Results for both testsets.

5.5 Analysis

Table 1 summarises the performance of the proposed model on the testset and the performance of the BiDAF baseline as stated in Moghe et al. [6]. For our model, the scores for both the testset and validation set are similar. Both perform improbably terrible on BLEU where this score does not seem to increase, whereas the ROUGE scores do. In addition, the ROUGE scores are significantly lower, this could indicate that the proposed model simply does not perform that well, or their implementation of the scores has not been used correctly. The latter is probably due to the fact that using their implementation compared to the PythonROUGE implementation, scores completely different and we could not get their implementation of the ROUGE-L score to work on the data due to memory limitations.

When analysing the dataset, it turns out that the background knowledge often contains incomplete sentences. This means that the model makes a representation of incomplete sentences. Something similar is done with the input. The previous answer is added to the next input to take history into account, but this is not always done very cleanly. It has occurred several times in the dataset that sentences are cut, resulting in input that starts mid-sentence. Even though this does not seem right, the model does not seem to suffer from these kind of sentences, but simply lacks complexity and length.

The model is capable of generating complex sentences that are often well-structured. The examples in this section are generated by the model of with the default parameters mentioned at the beginning of Section 5. Examples 1 and 2 are generated by the show generated responses that are coherent and that are good responses to the input. This is probably because this is an input sentence that occurs frequently, giving the model more data to train with this input.

```
  Example 1
Input: what is your opinion about the movie ?

Target 1: i think it was a great movie .

Generated 1: i think it was an excellent and heart
wrenching film

  Example 2
Input: what is your opinion about the movie ?

Target: i think it was a pretty funny funny movie .
jim carrey is a great actor

Generated: i think it was a great depiction of
realistic working class settings and environments ,
with great performances by robert de niro and
christopher walken .
```

Examples 2 and 3 demonstrate that the model has trouble with using the information from the right movie. This could be because no useful information was extracted from the weighted templates, and therefore the background knowledge does not have a strong influence when generating the responses.

```
  Example 3
Input: which scene did you like the most in the movie ?
the final fight scene between the green goblin and
spider man that was intense . spider - man has always
been an action packed movie .

Target: peter not only survives the wrestling match ,
he defeats bone saw in two minutes .

Generated: the minions follow scarlet to her castle and
kevin phones his fellow minions to come .
```

The model often generates sentences that do not fit the input, as shown in Example 4. The generated sentence is more likely to be an answer to "which scene did you like the most in the movie ?", which is a mix-up that happens often between these two inputs and their generated responses. This could be due to the inputs having too similar representations learned by the model.

```
  Example 4
Input: which is your favourite character in this ?

Target: my favorite character was frodo .

Generated: i like the one in which the future x - men
are fighting the <unk>
```

6 CONCLUSION AND FUTURE WORK

We proposed a model that weights background knowledge and use them as soft templates in order to generate human-like responses. In addition, our model allows the bilinear layer to be trained using the decoder loss.

Even though the use of the bilinear layer to incorporate background knowledge does improve the performance of the model significantly, the bilinear loss to boost learning does not. Our model does not outperform the baseline model from Moghe et al. [6]. This is most likely due to the fact that the bilinear loss is significantly lower than the decoder, which results the bilinear training almost solely on the decoder loss even when weighting them using a hyperparameter α, which may result in incorrect template selection. This could be the reason why the responses that were generated differed a lot from the target sentence.

Although our work did not outperform the baseline, we do believe the idea of weighting the background knowledge based on the decoder loss should be further explored. The goal of the current paper was to let the decoder loss boost the learning performance of

the bilinear, however, the method proposed in the paper resulted in the bilinear layer almost solely train using the decoder loss. Further work should try to look for a method to balance the two losses correctly.

Additionally, currently the task of the bilinear layer is to predict the ROUGE-scores between the input and the template and thus trained on the ROUGE-score between the target and the template. As the ROUGE-score is simply an estimate to the informativeness of the templates, it does not truly reflect which templates are good and which templates are bad, therefore, the target ROUGE-scores may not be a correct way of training the bilinear layer. A possible solution is to simply to predict a possibility distribution over the templates and use the cross-entropy with the correct template as the target. This may avoid errors caused by inaccurate estimates of the ROUGE-scores.

REFERENCES

[1] Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 152–161.

[2] Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2017. A knowledge-grounded neural conversation model. *arXiv preprint arXiv:1702.01932* (2017).

[3] William H Guss, James Bartlett, Phillip Kuznetsov, and Piyush Patil. 2017. Eigen: A Step Towards Conversational AI. *Alexa Prize Proceedings* (2017).

[4] Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155* (2016).

[5] Ryan Lowe, Nissan Pow, Iulian Serban, Laurent Charlin, and Joelle Pineau. 2015. Incorporating unstructured textual knowledge sources into neural dialogue systems. In *Neural Information Processing Systems Workshop on Machine Learning for Spoken Language Understanding*.

[6] Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M Khapra. 2018. Towards Exploiting Background Knowledge for Building Conversation Systems. *arXiv preprint arXiv:1809.08205* (2018).

[7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[8] Iulian V Serban, Chinnadhurai Sankar, Zhouhan Lin Saizheng Zhang, Sandeep Subramanian, Taesup Kim, Sarath Chandar, Nan Rosemary Ke, Sai Mudumba, Alexandre de Brebisson, Jose MR Sotelo, et al. 2017. The Octopus Approach to the Alexa Competition: A Deep Ensemble-based Socialbot. *Alexa Prize Proceedings* (2017).

[9] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models.. In *AAAI*, Vol. 16. 3776–3784.

[10] Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* (2015).