Attention Based Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion

Martin de la Riva, Sofia Herrero Villarroya, Tushar Nimbhorkar, Daniela Solis Morales
Univeristeit van Amsterdam
{martin.delarivaruiz, sofia.herrerovillarroya, tushar.nimbhorkar, daniela.solis.morales}@student.uva.nl

Abstract

For search engines to be able to formulate adequate query suggestions, it is essential to bear in mind the users' search intent and to be aware of the context. Sordoni et al. proposed a Hierarchical Recurrent Encoder-Decoder architecture to suggest queries using the previous queries as context. This paper presents A-HRED, an extension of this model using attention mechanism in order to improve performance. The result shows a broad idea of what attention can do but further work such as sample the query suggestions or hyper-parameter tuning could ground more the conclusions.

1 Introduction

Search engines use query suggestion to help users find the information they need. There are two types of query suggestions, auto-completions and reformulations. Auto-completions help users complete the query suggesting words while they are writing the query whereas reformulation suggestions are provided after one or more queries have been submitted to the search engine. The main focus of this paper are query reformulation suggestions.

For search engines to be able to formulate adequate query suggestions, it is essential to bear in mind the users' search intent and to be aware of the context. However, achieving context awareness is challenging due to data sparsity. Thus classical count-base methods become inaccurate.

Another hurdle is dealing with complex queries where the suggestions needed might not have been present in previous examples and so the context is not enough. Classical methods assume that the best recommendations have already been seen, but for rare queries, this might not be the case.

To introduce context in the query suggestion problem, Sordoni et al. propose in [Sordoni *et al.*, 2015] a Hierarchical Recurrent Encoder-Decoder (HRED) architecture. This model introduces context by encoding it and then decoding it for the suggestion. This representation of context embedded in a space avoids data sparsity since similar context are mapped close to each other.

However, is all of the context relevant? Mechanisms such as attention allow to focus on specific parts of the input. Based on the visual attention mechanism found in humans, which allows them to focus on certain region of the image, attention has been previously used to improve tasks such as image recognition. A similar approach can be taken for query suggestions based on context: we can attend over certain aspects of the context that are more important when generating a query suggestion. The attention mechanism could improve the query suggestion generation.

The main purpose of this paper is the replication of the HRED and the inclusion of the attention mechanism to answer the following research question: does attention improve the HRED query suggestion model?

The remaining of the paper is organised as follows. Section 2 gives a brief overview of related work; section 3 describes the HRED model; 4 describe how attention was adapted and implemented in the model; section 5 contains the experimentation and results obtained; and finally sections 6 and 7 outline the future work and conclusions.

2 Related Work

There are several desired characteristics for a model that is able to predict accurate query suggestions. The previous queries submitted by users during a session can be consider the context. Leveraging this context can reduce the ambiguity in the current query and therefore produce a better suggestion [Jiang *et al.*, 2014]. Sequence-to-sequence models used for Neural Machine Translation resemble this structure where the input sequence is mapped to a target sequence by means of an encoder-decoder structure [Sutskever *et al.*, 2014]. The model should also take into account the order of previous queries which indicates if the user is searching for a generalisation of the last search or it is searching for something more specific. Previous work on that [Boldi *et al.*, 2008] only exploits co-occurrence which is prone to data sparsity and has a difficulty to deal with rare queries.

The HRED architecture proposed by Sordoni et al. is a hierarchical encoder-decoder. Based on the sequence-to-sequence model, the input to the model would be a sequence of previous queries (the context). However, each query is a sequence itself. Therefore, the first level of the hierarchical recurrent encoder-decoder model encodes a query to a vector that compactly represents it while keeping the order of words. With this representation per query the sequence of previous queries is the input to the second level of the model, which will account for this order. The final level is the decoder to output the query suggestion.

To tackle the difficulty of rare queries the model is capable of producing synthetic suggestions [Jain *et al.*, 2011], that is, queries that have never been seen before but whose words are in the vocabulary.

Attention was first introduced in the Computer Vision field and has been widely used achieving positive results [Xu *et al.*, 2015]. However, it was with [Bahdanau *et al.*, 2014][Luong *et al.*, 2015] were attention was first introduced in Neural Machine Translation type of architectures.

3 HRED Model

The main objective is to build a model capable of producing synthetic and context-aware suggestions. This is achieved with a generative probabilistic model with a Hierarchical Recurrent Encoder-Decoder architecture.

The model is comprised of two parallel processes: the encoding and decoding of queries. On one hand, a Recurrent Neural Network (RNN) encoder that learns a compact order-sensitive encoding of a query up to that time. Then, a second encoder RNN keeps the context per session by learning a summary of the past queries. Finally, a decoder RNN is used to calculate a probability distribution on the space of possible queries given the encoded query. Then, this distribution is used to generate the next word in a sequence. This process lasts until the end-of-query symbol appears. The hidden state of the decoder is initialise with the context representation obtained from the second encoder to use it in the generation of the query suggestion. The next subsection describes the specifics of this model.

3.1 Model Architecture

The architecture of the model can be divided into three different Gated Recurrent Units (GRU) RNN [Cho *et al.*, 2014]: Query-Level Encoder, Session-Level Encoder and Next-Query Decoder.

Query-Level Encoder

This encoder receives as input a query $Q_m = \{w_{m,1}, ..., w_{m,N_m}\}$ where N_m is the length of the query and words are coded into word embeddings with dimension d_e. The RNN reads the word embeddings sequentially updating the hidden state. The final hidden state is a vector $q_m \in \mathbb{R}^{\text{enc}}$ storing order-sensitive information about all the words in the query. The query is now represented with this vector. The hidden state is updated according to:

$$d_{m,n} = GRU_{query}(d_{m,n-1}, w_{m,n}), n = 1, ..., N_m. \quad (1)$$

Session-Level Encoder

This encoder receives as input the sequence of query representations $q_1, ..., q_m$ and repeats the same process as the RNN but at a query level. The final recurrent state $s_m \in \mathbb{R}^{\text{sess}}$ encodes the queries that have been processed up to that point keeping their order. The hidden state is updated according to:

$$s_m = GRU_{sess}(s_{m-1}, q_m), m = 1, ..., M. \quad (2)$$

Next-Query Decoder

The decoder predicts the next query Q_m given the previous queries as context. This information is obtained by initialising its recurrent state with s_{m-1} as follows:

$$h_{m,0} = tanh(D_0 s_{m-1} + b_0) \quad (3)$$

$$h_{m,n} = GRU_{dec}(h_{m,n-1}, w_{m,n}), n = 1, ..., N_m. \quad (4)$$

Then, each recurrent state $d_{m,n-1} \in \mathbb{R}^{\text{dec}}$ is used to compute the probability of the next word $w_{m,n}$ given the previous words and queries as:

$$\begin{aligned} P(w_{m,n} = v | w_{m,1:n-1}, Q_{1:m-1}) = \\ = \frac{\exp o_v^T \gamma(h_{m,n-1}, w_{m,n-1})}{\sum_k \exp o_k^T \gamma(h_{m,n-1}, w_{m,n-1})}. \end{aligned} \quad (5)$$

Where the o_v is the embedding of the vocabulary used to learn to position related vocabulary words closer to the context information encoded by γ. The linear transformation γ on $d_{m,n-1}$ and the previous word $w_{m,n-1}$ is defined as:

$$\begin{aligned} \gamma(h_{m,n-1}, w_{m,n-1}) = \\ = H_{output} h_{m,n-1} + E_{output} w_{m,n-1} + b_{output}. \end{aligned} \quad (6)$$

The γ function receives words and embeds them in an additional embedding space. Therefore, throughout the whole model there are three embedding spaces so the input, the vocabulary words and the previous words are represented in different spaces achieving as much expressive power as possible. However, training for three spaces implies longer training time since there are more weights to learn. This is a trade-off between the expressiveness and time-complexity.

3.2 Learning

The RNNs will learn by maximizing the log-likelihood of a session S using back-propagation through time (BPTT) and optimizing with Adam. The parameters to learn are the weights for each GRU RNN, the three embedding layers (input, o_v and γ function) and the fully connected layer weights to project the session state to the decoder dimensions. The loss of a session is:

$$\begin{aligned} \mathcal{L}(\mathcal{S}) &= \sum_{m=1}^{M} logP(Q_m | Q_{1:m-1}) \\ &= \sum_{m=1}^{M} \sum_{n=1}^{N} logP(w_{m,n} = v | w_{m,1:n-1}, Q_{1:m-1}). \end{aligned} \quad (7)$$

For the training and validation stage equation (6) will be calculated using the previous word $w_{m,n-1}$, the decoders previous output, as the next input. The initial states for the GRU are the null vector except for the decoder that uses the session state.

Table 1 specifies the values for different parameters of the model for the training process. The dimension of the layers was reduced in comparison with [Sordoni *et al.*, 2015] (from *enc* = 1000 to 500 and from *sess* = 1500 to 750) due to lack of computational resources.

Table 1: HRED parameters.

Parameter	Value
Optimizer	Adam
Learning Rate	0.002
Dimensionality of embedding layer (e)	300
Dimensionality of query encoder RNN (enc)	1000
Dimensionality of session encoder RNN (sess)	1500
Dimensionality of decoder RNN (dec)	1000
Batch size	60

3.3 Generation

Beam-search algorithm is used to produce a query suggestion: iteratively consider a set of k best words as candidates and then extend each of them by sampling the most probable k words given the distribution generated by the decoder RNN. Process ends when k well-formed queries ending with the end-of-query symbol are obtained.

Given two queries submitted by the user, each query is fed to the encoder RNN to obtain the query vector q_1 and q_2. Then, these two are fed to the session-level RNN to obtain the recurrent states s_1 and s_2. At this point the information has been encoded to proceed with the generation of the context-aware suggestions. The decoder RNN uses the last session-level hidden state, s_2, as the initial recurrent state and using it the suggestions are sampled by means of a beam-search. For each word, the probability to be the first in the suggestion is calculated using equation (5). The k words with the highest probability are added to the beam of size k. The next decoder recurrent state is computed using the previous state and the sampled word. The process repeats until the end-of-query symbol is sampled.

4 Attention

Broadly speaking attention consist on defining how much of each input state affects each output. To achieve this, a score is calculated given an output for each input. This score is based on the context of the input. Therefore, the score weighs each input and combines them so that the output is affected selectively. However, this process requires to scan the inputs for every output to obtain the scores and then re-scan the important inputs to produce the actual output. Counterintuitively, it does not save time, as the human visual attention does, but it achieves a good performance. It also avoids forcing the network to encode all the information in the vector by allowing it to refer back to the input and decide which one to use.

Therefore, the main idea of attention is to determine which inputs are relevant to predict the query and pay attention to those by weighting them more. Attention could be applied to word-level and determine which words are relevant; or could be applied to the query-level and determine which queries are relevant. Given that the suggestion is generated per word the most natural approach for attention is the former and thus it is the one implemented.

The implementation of attention is based on the original paper that introduced it [Bahdanau *et al.*, 2014] and in [Luong *et al.*, 2015]. The input, in this case a word embedding, is represented in a compact form a_j that encapsulates the information that is used to determine how important it is. This annotation $a_j \in \mathbb{R}^{2enc}$ is a summary of the preceding and following words in order to capture the context for each word. The original paper proposes a bi-directional RNN to get this representation. For our model, this is implemented with a bidirectional RNN with GRU cells. An annotation is the concatenation of the forward d_f and backward states d_b of the bidirectional RNN:

$$a_j = [d_{f,j}; d_{b,j}]. \tag{8}$$

The original implementation in [Bahdanau *et al.*, 2014] was done in a Neural Machine Translation model for which the inputs were the words of a given text. The attention implemented was calculated over the whole text to determine which words were important for the translated output. Given the scenario of this paper, attention is adapted to pay attention over the words for a query and not over all of the words in the session. Therefore, the annotations are calculated per query and then the states of the bidirectional RNN are reset after a query. On the contrary, the original paper used the same states for the whole text.

The score $score(h_{n-1}, a_j)$ determines how well the word annotation and the output n match by using the hidden state of the decoder h_{n-1} before emitting output n. Then, this score is used to weight each query for output i:

$$c_i = \sum_{j=1}^{N} softmax(score(h_{i-1}, a_j)) * a_j \tag{9}$$

Being N the number of words up to time i. Thus, there is a context vector c_i per output i.

The scoring function is learned with a fully connected layer with weight $W_{score} \in \mathbb{R}^{\text{dec x enc}}$ where dec is the decoder dimensions and enc is the encoder dimensions:

$$score(h_{i-1}, a_j) = h_{i-1}^T W_{score} a_j \tag{10}$$

The context vector is introduced in the original paper as a condition when calculating the probability for a word. Consequently, the HRED architecture with attention (A-HRED) would include this vector as:

$$P(w_{m,n} = v | w_{m,1:n-1}, Q_{1:m-1}, c_i) \tag{11}$$

In the implementation the context vector is concatenated to the output of the decoder.

5 Experimentation

This section describes the experimentation carried out to answer the research question: does attention (A-HRED) improve the performance of the HRED model?

Section 5.1 explains how the data set has been compiled for training and validation. Section 5.2 explains the metrics implemented. Sections 5.3 and 5.4 describe the experiments carried out and the experiments.

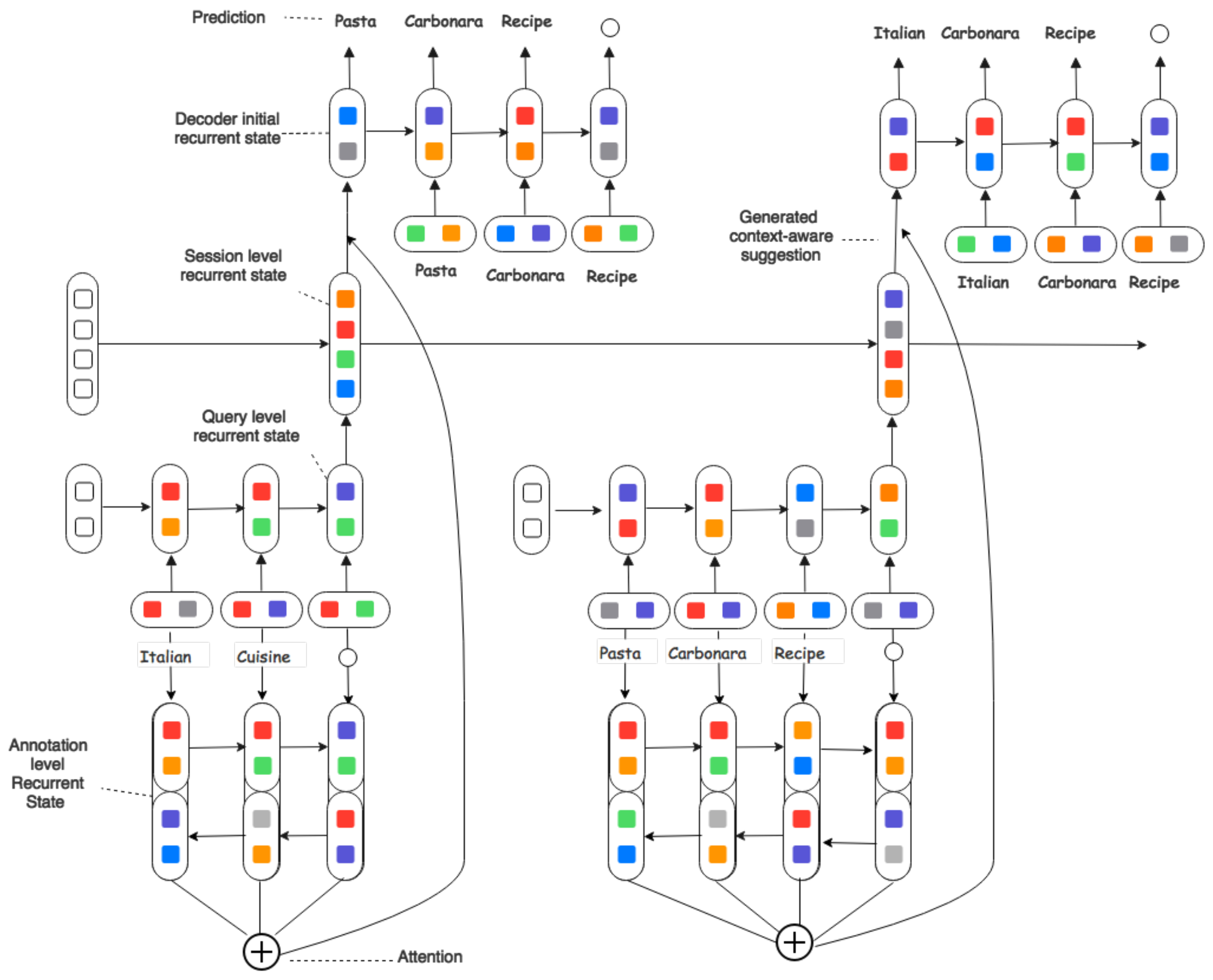
Prediction
Pasta
Carbonara
Recipe
Decoder initial recurrent state
Pasta
Carbonara
Recipe
Session level recurrent state
Query level recurrent state
Italian
Cuisine
Annotation level Recurrent State
Attention
Italian
Carbonara
Recipe
Generated context-aware suggestion
Italian
Carbonara
Recipe
Pasta
Carbonara
Recipe

Figure 1: RNN pipeline for the A-HRED model where each arrow is a non-linear transformation and the circle symbol represents the end-of-query symbol. First, the user queries *Italian Cuisine* and then *Pasta Carbonara Recipe*. When training, the model encodes *Italian Cuisine* and gets its annotations. It then updates the session-level recurrent state and uses attention to determine the output that maximizes the probability of seeing the following query *Pasta Carbonara Recipe*. This process is repeated for all queries in the session. For testing, a contextual suggestion is generated by encoding the previous queries and determining the attention, then updating the session-level recurrent states accordingly and finally sampling a new query. In the example, the generated contextual suggestion is *Italian Carbonara Recipe*.

5.1 Data set

The data set used is the AOL data set which is composed of query searches from 1st of March 2006 to 31st May 2006. The data set has been pre-processed as the original authors did, which can be found in their Github [1]. The data was divided into sessions by considering the end of one after 30 minutes of the user being idle [Jansen Bernard *et al.*, 2007]. Table 2 specifies how is the AOL data set split for the model. The data is available in the github repository for this project.

Given that there are queries with different lengths each example is padded to a fixed maximum length of 10. Then, the implementation uses a dynamic length RNN that handles the optimization only on the non-padding symbols. The loss and the metric will also only account for the non-padding symbols.

To handle the sessions correctly the batches are dynamic to ensure that a session is not split amongst batches. This is important since the session-encoder will restart the states for every batch, which is every session.

The model was validated with the same set of 100 batches

Table 2: Dataset information.

Type	Number of sessions	Sample period
Background	1937785	1/03/2006 - 30/04/2006
Training	448302	1/05/2006 - 15/05/2006
Validation	291559	16/05/2006 - 24/05/2006
Test	180986	25/05/2006 - 31/05/2006

[1] `https://github.com/sordonia/hred-qs`, visited on 5th Dec 2017.

during the training process.

5.2 Metric

Two simple metrics were implemented to measure the performance in the validation data set:

- **Accuracy over the non-padding symbols**: counts how many words of the query are predicted and if they are in the correct position.
- **Words predicted over the non-padding symbols**: counts how many words are predicted correctly, regardless of the position in which they are predicted. This metric measures in a simplistic manner how well the model captures the context: the model predicts the right word, just not in the right position.

However, these metrics are very simple given the stage of the experimentation that we were in. For further experimentation, for instance, when the generation stage is performed, more specific metrics would be required, such as Mean Reciprocal Rank (MRR) when tested in a learning-to-rank approach. Nevertheless, the metrics proposed allow to get some insights of the model to determine its correctness and identify issues.

5.3 Experiment 1

The goal of this experiment is to determine the correctness of the model conceptually by overfitting it with the same batch. If the model overfits then one can assume that the logic of the implementation is correct. The experiment was carried out for both models. Table 3 shows the results.

Table 3: Results for experiment 1.

	Iterations	Loss	Accuracy	Words predicted
HRED	4000	0.81	0.78	0.81
A-HRED	4000	4.38	0.83	0.84

Running for 4000 iterations already shows that the models are overfitting since the loss is very low and the metrics are high (they were already above 0.7 accuracy after 1000 iterations). HRED shows that it would converge faster than A-HRED which could be because A-HRED is more complex. From these results we can confirm that the models are logically correct. In addition, it seems that A-HRED could perform better although these results are preliminary and the following section describes the experiment to determine this.

5.4 Experiment 2

This experiment aims to fully compare both models by training them with the best hyper-parameters values. However, an extensive search of those could not be carried out mainly because of the lack of computational resources - the dimensionality of the RNN already had to be decreased to a suboptimal value, assuming that Sordoni et al. reported the best hyper-parameters values. Therefore, these results are not conclusive but give a general insight.

Table 4 presents the results for this second experiment and Figure 2 shows the accuracy curve. It can be seen that both models are performing quite same on metric of accuracy. The loss for the A-HRED model is higher than for the HRED meaning that it is further away from convergence. It is possible that if the A-HRED model is let run for more iteration it will get the same accuracy than the HRED model but given that it is more complex, it takes more time to converge. Unfortunately due to the RAM limitations on SurfSara we could not do the full training loop which was supposed to be around 4 Million iterations.

Table 4: Results for experiment 2.

	Iterations	Loss	Accuracy	Words predicted
HRED	37000	18.45	0.40	0.41
A-HRED	37000	44.32	0.38	0.39

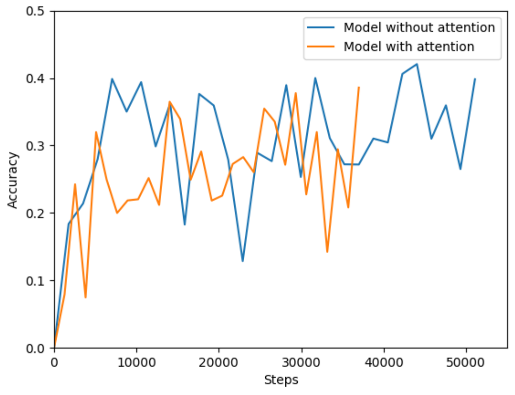

Figure 2: Accuracy vs. Number of iterations for both model (the model without attention ran for 50000 iterations while the model with attention crashed at 37000 due to memory issues).

5.5 Analysis of the results

This section outlines the observations obtained from the experimentation carried but with both models.

During training, accuracy (on validation set,Figure 2) has had large oscillations instead of having a stable increase. This could be due to the fact that it may not be an appropriate metric for this task. Since the suggestions are open, not having perfect accuracy in the validation set means that the network is not fully learning the validation queries however it could be suggesting similar ones (this is in the lines of the the idea proposed by Sordoni et al. about the ability to generate synthetic queries). In an attempt to solve this, we also had the words predicted metric which has shown to be usually a bit higher that the accuracy. This means that the words are at least in the query. While this metrics could be useful for getting some insight of the training, we realise that what it is necessary to measure is the generated suggestions. In order to automate this, a learning-to-rank approach should be used, otherwise, it would imply manually checking each suggestion.

In addition, the results are not comparable to Sordoni et al. since we could not use the same parameters as they were using due to computational restrictions. We initially had a maximum query length of 50 but this was reduced to 10 to generate smaller Tensors that could be allocated in the RAM memory. We expected this change to not have a great impact in the data set quality since the medium query in the AOL data set has length 2 and already queries of length 25 have a low probability as shown in Figure 3. Nevertheless, there are queries as long as 245 words which are cut down to 10. Also, the dimensionalities of the RNNs had to be reduced compared to the original implementation for the same issues, and this might have had an impact in the performance.

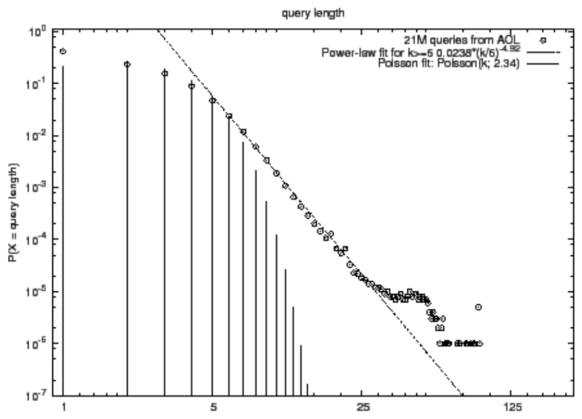

Figure 3: Poisson and Power-Law fit for the probability of the query length in the AOL data set (source).

Finally, it is important to remark that a hyper-parameter search is required to obtain comparable results. Due to the complexity of the model, this is quite a computationally expensive task that would be interesting to perform for future work.

6 Future work

This project has given some first insights of what attention mechanisms can achieve in hierarchical encoder-decoders architectures. However, the results are not conclusive and exhaustive experiments should be carried out in the future to obtained more grounded conclusions. This project has left some clear future work to fully complete it. One first step would be to implement the generative part of the model (integrate beam search with the full architecture) to fully compare both models in this sense. Regarding the experimentation, it would be required to perform parameter tuning to get a fair comparison. Also, further experimentation with different scenarios would be interesting in order to test the capabilities of A-HRED - for instance, using a learning-to-rank set up as proposed in Sordoni et al.

The attention implemented for A-HRED was at word-level to decide the important words, but, as mentioned, it could also be at query-level to select the important queries in the session. It would be interesting to compare both to each other or further more using both determine the most effective one. Also, the attention scored the inputs with respect to previous queries but other possible approaches could be based on the user behaviour. Given that the AOL data set contains information on the clicks, this data could be use to determine the important previous queries.

7 Conclusions

This paper has presented an extension of the work by Sordoni et al. in [Sordoni *et al.*, 2015] where a Hierarchical Recurrent Encoder-Decoder (HRED) architecture for query suggestion was introduced. The model has been extended to the A-HRED model that introduces an attention mechanism over the words of the queries.

The implementation of both models has proven that the HRED model was already quite complex and that introducing attention resulted in a more complex model, as seen in the training process since it takes longer to converge. However, if we were able to fully analyse the capabilities of attention maybe it would be possible to reduce the complexity of the model and rely more in attention mechanism. The final implementation still requires the generation module (beam search) to fully compare both models and get insights of the generation capabilities and how these are affected by attention.

References

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

[Boldi *et al.*, 2008] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. The query-flow graph: model and applications. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 609–618. ACM, 2008.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[Jain *et al.*, 2011] Alpa Jain, Umut Ozertem, and Emre Velipasaoglu. Synthesizing high utility suggestions for rare web search queries. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 805–814. ACM, 2011.

[Jansen Bernard *et al.*, 2007] J Jansen Bernard, Amanda Spink, Chris Blakely, and Sherry Koshman. Defining a session on web search engines: Research articles. *Journal of the American Society for Information Science and Technology*, 58(6):862–871, 2007.

[Jiang *et al.*, 2014] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. Learning user reformulation behavior for query auto-completion. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 445–454. ACM, 2014.

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

[Sordoni *et al.*, 2015] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM, 2015.

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

[Xu *et al.*, 2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.