Seminar Report: Deep Learning Sequence Modelling (Natural Language Processing)

Neha Das
Technical University of Munich
neha.das@tum.de

Abstract

Recent experiments with deep learning techniques in the field of sequence modelling tasks in Natural Language Processing (NLP) such as machine translation and text summarizing have been quite successful and have produced improved results over classical methods. This work will take a look at the various deep learning architectures and constructs used to model sequences and aid tasks that involve processing and/or producing sequence data, especially in context of NLP. It would additionally explore in detail, an application of sequence-to-sequence NLP - abstractive text summarization - with particular emphasis on methods from Nallapati et al. (2016)

1 Introduction

Natural Language Processing applications often deal with sequential data. While in applications such as Parts of Speech tagging or Named Entity Relationship modelling, the output is often a single value corresponding to the currently processed token in the sequence, other applications in this domain such as machine translation, video captioning and text summarization output a sequence whose constituents do not necessarily have a one to one relationship with the input.

Recently, deep learning methodologies have successfully been used to model many of the applications in the latter category. These models are often referred to as "sequence to sequence" models and have provided improved results for machine translation (Sutskever et al., 2014; Cho et al., 2014a), abstractive text summarization (Rush et al., 2015; Nallapati et al., 2016), video captioning (Venugopalan et al., 2015) and other such applications over other methods such as statistical phrase-based systems (Banko et al., 2000), which have a large memory footprint because of the huge phrase table that is created in the process or rule-based systems (Cohn and Lapata, 2008), which are domain specific and need a lot of linguistic expertise and manual effort for rule creation.

Deep learning models often utilize an encoder-decoder approach where they encode the input to a fixed-length representation (Cho et al., 2014a) and feed it to the decoder and generate the output sequence. Over-time however, additional constructs such as "attention" (Bahdanau et al., 2014) and incorporation of additional features along with the plain input (Nallapati et al., 2016) have facilitated in creating models that give more accurate results.

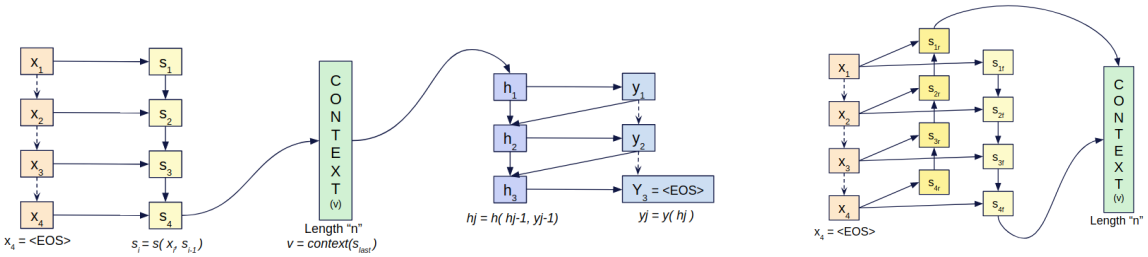
The aim of this body of work is to introduce all these concepts and then focus our attention on modelling a particular application - abstractive text summarization - from the paper by Nallapati et al, (2016)

2 Encoder-Decoder Architecture

The encoder-decoder architecture forms the basis of nearly all sequence modelling techniques in deep learning. Most notably, this approach was used in (Sutskever et al., 2014), for machine translation from English to French texts. In the early versions of this model, the sequential input is fed to an encoder deep

neural network (DNN) which produces a fixed length context. This fixed length context vector is then fed to the decoder, another neural network, which produces the output.

Since the encoder and decoder each process or output sequential data, they are often chosen to be Recurrent Neural Networks (RNN) (Sutskever et al., 2014; Bahdanau et al., 2014), as you can see in Figure 1(a). However, Convolutional Neural Networks (CNN) (Rush et al., 2015) and Gated recursive Convolutional Neural Networks (Cho et al., 2014a) have also been experimented with as substitutes for RNN encoders. A limitation of using CNNs could be that while they preserve spatial relations, they do not preserve the ordering in sequences.



(a) (b)

Figure 1: (a) An encoder-decoder architecture with an RNN encoder and an RNN decoder. x_i is the encoder input at time step i of the encoder, s_i is the hidden state at time step i calculated by function s, v is the context vector of length n, h_j represents the hidden state at time step j calculated by function h and finally, y_j is the output at time step j of the decoder and is calculated using y. (b) A bidirectional encoder. The layer with hidden units s_f is for modelling forward dependencies, and the one with hidden units s_r models backward dependencies.

2.1 Encoder-Decoder Framework with RNN

As described in Figure 1(a), the encoder RNN works on the input sequence $X = \{x_1, ..., x_n\}$ and produces a hidden state s_i at each timestep i where s_i is the function of the input at timestep i and the hidden state from the previous time step s_{i-1}. The context vector v is the function of the last hidden input s_n and is fed into the decoder.

The decoder consists of hidden states $H = \{h_1, ..., h_m\}$, that are each a function of the hidden state from the previous timestep h_{j-1} and the output from the previous timestep y_{j-1}. The first hidden unit h_0, however takes context vector v as an input. The output y_j at each timestep is extracted from a probability vector over all the target vocabulary calculated by a softmax function (Cho et al., 2014a) that uses the hidden state h_j ($\{W_t, b_t\}$ are learnable parameters):

$$P(y_{j,t} = 1|v, y_1, ..., y_{j-1}) = \frac{exp(W_t h_j + b_t)}{\sum_{t'=1}^{T} exp(W_{t'} h_j + b_{t'})} \tag{1}$$

How exactly the output y_j at each timestep j is inferred, will be explored in a later Section. During training, however, we only calculate the probabilities of the words that appear in the ground truth sequence and use them to maximize the conditional probability described below:

$$P(y_1, ..., y_m|x_1, ..., x_n) = \prod_{j=1}^{m} P(y_j|v, y_1, ..., y_{j-1}) \tag{2}$$

The goal of a sequential model is to maximize the conditional probability of the obtained sequence, given the input sequence. With encoder-decoder modelling, however, this probability could be decomposed (Sutskever et al., 2014) like in the above equation. In effect, of course, instead of maximizing this conditional probability, we aim to minimize its negative log-likelihood during our training process.

2.2 Modelling dependencies in both directions - Bidirectional RNN

The RNN model introduced in section 2.1 models dependencies in input sentences in a single direction. This is because the hidden state s_i in the encoder depends on only the previous hidden input s_{i-1}, but not on the next one (s_{i+1}). As a result, the hidden units cannot derive any context from the next token, even though the current token significantly relies or even derives its meaning from the next one. Bidirectional RNNs (Schuster and Paliwal, 1997) resolve this issue. As shown in Figure 1(b), bidirectional RNNs contain an additional layer of hidden units with reversed connections so that they can model dependencies in the backward direction.

2.3 Overcoming Limitations of Vanilla RNNs

The RNN model from section 2.1, also known as the vanilla RNN, generally suffers from a limitation known as the vanishing/exploding gradient problem that exists because the weights of an RNN must be adjusted by backpropagating through time (BPTT). This makes vanilla RNNs unsuitable for modelling long term dependencies (Britz, 2015).[1]

These issues can be dealt with by using a modified version of vanilla RNN architecture such as Long Short Term Memory (LSTM) (Gers et al., 1999; Hochreiter and Schmidhuber, 1997) or Gated Recurrent Units (GRU) (Cho et al., 2014b).

2.4 Limitations of the Encoder-Decoder Model with Fixed Sized Context

As pointed out in the paper by Bahdanau et al. (2014), using a fixed sized context vector, like in the models introduced in Section 2, can have a severe limitation - it is hard to model long sequences with such architectures since a fixed sized context vector cannot appropriately represent a sequence as it keeps getting longer. Additionally, since the context vector is derived from the last input hidden unit, it may not be laying due emphasis on the earlier parts of the sequence. As a consequence, the quality of the output produced by the network suffers when the sequence length is increased. This has been empirically observed by Cho et al.(2014a). A solution to this predicament is to discard the fixed sized context vector and use the concept of attention instead (Bahdanau et al., 2014).

3 Attention Mechanism

The attention mechanism was first used in a sequential modelling setting by Bahdanau et al.(2014) to overcome the limitations of a fixed sized context vector. It involves the creation of a context vector c_j which serves as an input for each decoder hidden state h_j. Each of these context vectors are computed as a weighted sum of all of the input hidden states $S = \{s_1, ..., s_n\}$. The weights $\{a_{i,j}\}_{i=1,j=1}^{n,m}$ intuitively describe the influence of input x_i on the output y_j. These concepts are illustrated in Figure 2(a).

The attention weights are calculated as shown in equation 3. Here $e_{i,j}$ is an alignment model that scores the alignment between inputs around position i and outputs around position j and depends on the previous output hidden state h_{j-1} and the input hidden state s_i. The coefficients of the alignment model e are learnable parameters [2]. This means that the network learns to provide attention to influential parts of the input while discarding the non-essential components when it predicts an output.

$$a_{i,j} = \frac{exp(e_{i,j})}{\sum_{i'=1}^{n} exp(e_{i',j})} \; \textit{where} \; e_{i,j} = e(s_i, h_{j-1}) \tag{3}$$

[1] See Part 3 of Danny Britz's tutorial on Recurrent Neural Networks (Britz, 2015) for a thorough treatment of BPTT and vanishing gradients

[2] The notation is tweaked from the original formulation in Bahdanau et al. (2014) slightly in order to avoid conflicts with the established notation

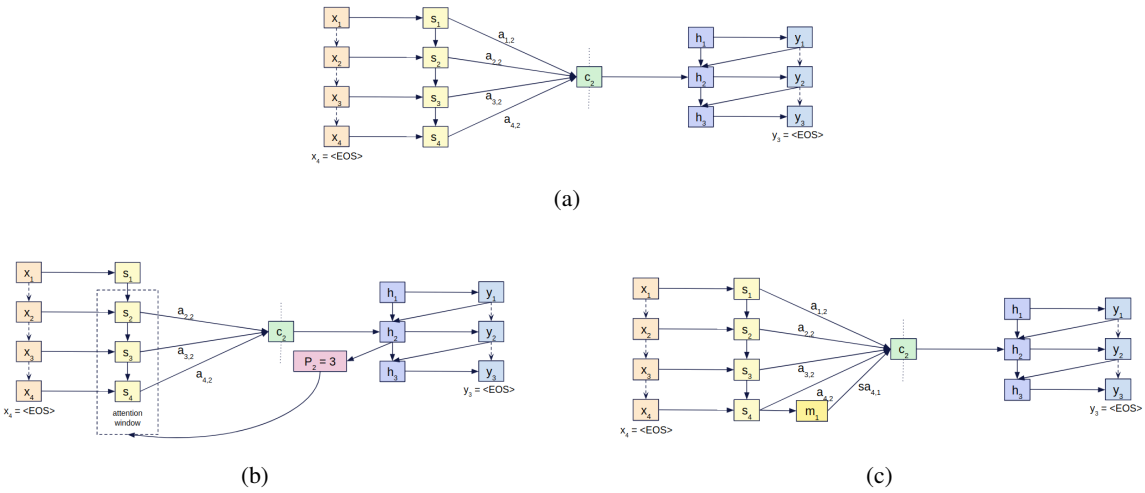


Figure 2: (a) Attention Mechanism. x_i and y_j represent the input and outputs at timesteps i and j respectively (b)Local Attention Mechanism. The attention window in this example is focused around position 3 of the input sequence. This position is outputted by the decoder via a predictive alignment function (Luong et al., 2015). (c) Hierarchical Attention Mechanism. The attention mechanism is present at both the word and sentence level (Nallapati et al., 2016).

3.1 Local Attention

A disadvantage of the attention mechanism described above is that the computational expense of the attention weight calculation increases with the increase in input sequence length. This can be handled by calculating the attention weights for an output y_j at decoder position j over a fixed window localized around a single position p_j in the source sequence (Luong et al., 2015) (See Figure 2(b)). p_j can be calculated as the result of a function taking the decoder hidden state at timestep j as an input.

3.2 Hierarchical Attention

In text summarization in particular, the input tends to be multi-sentencial. As a consequence, in addition to words, it might be beneficial to lay due emphasis upon the key sentences as well. Heirarchical attention mechanism (Nallapati et al., 2016) thus models attention both at word level and sentence level (See Figure 2(c)). To incorporate attention at both levels in the context for an output step, word/token level attention at each step is weighed by the attention weight of the sentence it is a part of.

4 Inferring output from the Decoder - Beam Search

Recall from Section 2.1 that the decoder outputs the probabilities over all the words in the target vocabulary. While in training, output y_j is chosen to be the one in the reference summary, at inference time we would like to produce a sequence $\{y\}_{i=1}^{m}$ that maximizes the conditional probability $P(\{y\}_{i=1}^{m}|\{x\}_{i=1}^{n})$. The foolproof way of achieving a sequence that satisfies the above condition, is however, by conducting an exhaustive search through the target space - a task of combinatorial complexity, and therefore virtually non-scalable and impossible for tasks with large target vocabularies. Another technique at the opposite end of the extremum is the greedy approach - at each step j of inference, the decoder chooses the word from the target vocabulary with the highest conditional probability $P(y_j|v, y_1, ..., y_{j-1})$.

The most frequent approach however, for inferring from sequential modelling architectures is beam search (Koehn, 2004). In this approach, at each step j of inference, the decoder chooses the first b sequences $\{y\}_{k=1}^{j}$ from the target vocabulary that maximize the joint probability $P(y_j, y_{j-1}, ..., y_1|v)$, where b is the size of the beam, such that the sequence $\{y\}_{k=1}^{j-1}$ is a sequence from the b options at step

$j-1$ of the decoder and the output $y_j \, \epsilon \, target\_vocabulary$

5 Application - Abstractive Text Summarization

Automatic Text Summarization is a well known application of sequence modelling and has been used and refined through research across a variety of domains - medical, legal and print media. Methods tackling this problem are usually classified into two broad categories - extractive summarization and abstractive summarization. Extractive techniques create a summary from a long text by concatenating a selection of sentences (key phrases) from the text itself. The aim of abstractive text summarization, on the other hand, is to summarize the main ideas of the document or input text, potentially using words or phrases unseen in the source document but possibly relevant to the compression of its key concepts. As expected, this is a harder problem to solve.

In the recent years, however, deep learning techniques have achieved state-of-the-art results for abstractive text summarization (Rush et al., 2015; Nallapati et al., 2016). Many of the concepts outlined in the previous sections have been incorporated in the architectures implemented in the aforementioned papers. This section will describe one such architecture and subsequently the results from the paper "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond" by Nallapati et al (2016).

5.1 Datasets

The deep learning architectures introduced in this paper have been trained and/or evaluated using three different corpora. A description of these datasets are provided in Table 1.

Dataset	**Input Texts**	**Reference Summary**	**Purpose**	**Added Features**	**Size**
Gigaword	Newswire text data from four different international sources	Headlines	Training and evaluation	Yes; provides the possibility to extract POS and NER tags and TF-IDF statistics	3.8 M training pairs, 2000 validation pairs and 2000 test pairs
DUC (2003,2004)	News articles from New York Times and Press Wire services	4 reference human generated summaries	Only evaluation	No; only provides the possibility of extracting word tokens	(2003) 624 pairs, (2004) 500 pairs
CNN/Daily Mail (modified)	Stories from CNN and Daily Mail websites	Summary bullets: multi-sentencial form	Training and evaluation	Yes; provides the possibility to extract POS and NER tags and TF-IDF statistics	286,817 training pairs, 13,368 validation pairs and 11,487 test pairs

Table 1: Training/Evaluation Corpora.

5.2 Architecture

The neural networks implemented and evaluated in this paper consist of combinations of the concepts introduced in the above sections. The most basic of these is an encoder-decoder model with attention, a bidirectional GRU encoder and a unidirectional GRU decoder. This forms the basis of all the other neural architectures that are evaluated in this paper. Some other constructs that have been used to create and evaluate more sophisticated models are hierarchical attention (Section 3.2) and a novel technique called the switching generator-pointer model (explained in the following subsection).

5.2.1 Switching Generator-Pointer Model

Often, the words or phrases in a test document may be central to the summary but are rare or unseen during the training time, and as such excluded from the target vocabulary which is fixed at the time of inference. Instead, the decoder emits a $< UNK >$ token in place of these out-of-vocabulary (OOV) tokens. However, the presence of such tokens in the output summary makes it illegible.

This paper handles the above predicament by using a switch mechanism (see Figure 3) between two decoder components - a generator that can generate a new token from the vocabulary at time step j, given the output hidden state h_j, and a pointer that produces the position of the token in the source document that the decoder would like to point to instead of outputting a $< UNK >$ placeholder.

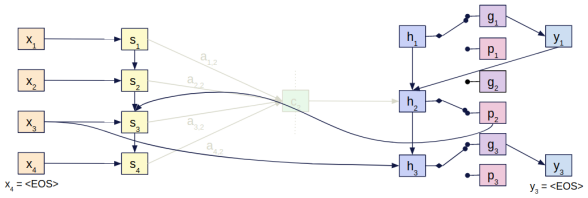


Figure 3: Switching mechanism between generator and pointer.

The state of the switch (sw) is modelled using the following probabilistic expression:

$$P(sw_j = 1) = sigmoid(sw(h_j, c_i, y_{j-1})) \tag{4}$$

The output of the decoder is modelled by the following expression:

$$logP(Y|X) = \sum_{j=1}^{n} (g_j log\{P(y_j|\{y\}_1^{j-1})P(sw_j)\} + (1 - g_j) log\{P(p_j|\{y\}_1^{j-1})(1 - P(sw_j))\}) \tag{5}$$

where, $P(Y|X)$ is the probability of sequence $Y = \{y_1, ..., y_n\}$, g_j is an indicator function and is set to 0 whenever the word at position j in the reference summary is OOV with respect to the target vocabulary, $P(y_j|\{y\}_1^{j-1})$ (See equation 1) is the decoder output probability using the generator given the previous outputs, and $P(p_j|\{y\}_1^{j-1})$ is probability of the output of the pointer, given the previous outputs. p_i is calculated as $argmax_i(a_{i,j})$ where $a_{i,j}$ is the attention weight for output j and input i and is calculated using equation 5.

5.3 Inference and Training Details

The network decoder uses beam search with a beam of size 5 to infer output words. The summary size is limited to 30 words, this being the size of the largest summary in the sampled validation set from the Gigaword corpus. Further, the training and evaluation processes utilize The Large Vocabulary Trick (Jean et al., 2014) - a technique that creates the decoder vocabulary for a minibatch during the training time by picking up the most frequent words from the target dictionary in addition to the words in the minibatch's source documents - in order to reduce the size of the decoder's soft-max layer. This not only helps in cutting down the computational complexity of the decoder, but also increases the convergence rate of the training process, by concentrating on words essential to the given summary. In addition to this, the best performing architecture (on the Gigaword corpus) takes linguistic features such as the TF-IDF scores, Name-Entity Relationship tags and Parts of Speech tags as an additional input along with the word embedding vector for each token to compose a feature rich encoder. As observed in the next section, the incorporation of additional features improve the summarization results. All the models were trained using Adadelta with an initial learning rate of 0.001 over a batch of 50 pairs and random-shuffling every epoch. To regularize the training process, early stopping was used based on the validation sets along with gradient clipping.

5.4 Evaluation

The authors of this paper used two different metrics are used to evaluate the summarization results produced by the trained models - The first, ROUGE, is used to evaluate the quality of the output and its similarity to the reference summary(-ies) from the dataset. ROUGE-F1 is preferred over ROUGE-recall by the authors since Recall scores do not penalize longer summaries, but Recall scores have also been calculated for comparison with state-of-the-art techniques. The second metric, Src. copy rate, calculates the percentage of words in the output summary that were part of the source document. Thus, this metric measures the abstractive ability of the model. The models introduced in this paper were evaluated over the three datasets introduced in Section 5.1:

5.4.1 Gigaword Corpus

As noted before, the Gigaword corpus is a large collection of annotated news articles from 4 international sources. The annotations in the Gigaword corpus are used by some of the models (Table 2) to extract features additional to the word embedding vectors that are inputted to the network. Table 3(a) summarizes the models that were trained and evaluated on the Gigaword Corpus.

Models	**Input**	$\|X_s\|$	HA_{bool}	SGP_{bool}	**Remarks**
words-lvt2k-1sent	Word Embeddings	One	False	False	This is the baseline architecture (See table description)
words-lvt2k-2sent	Word Embeddings	Two	False	False	Improved Results over baseline. Inclusion of more sentences in the input noted to degrade the result.
words-lvt2k-2sent-hieratt	Word Embeddings	Two	True	False	Noted to learn the relative importance of the two sentences
feats-lvt2k-2sent	Word Embeddings + TF-IDF Scores + POS Tags + NER Tags	Two	False	False	Incremental gains over words-lvt2k-2sent
feats-lvt2k-2sent-ptr	Word Embeddings + TF-IDF Scores + POS Tags + NER Tags	Two	False	True	This is the best performing model for this corpus as noted in Table 3(c)

Table 2: Models trained and evaluated on the Gigaword corpus. The baseline architecture, *words-lvt2k-1sent*, consists of an attentional encoder-decoder with a bidirectional GRU encoder and a unidirectional GRU decoder. The Large Vocabulary Trick is used to build and train on the target vocabulary. $|X_s|$ is the number of sentences used from the reference summary for training the model. HA_{bool} indicates the use of Heirarchical Attention. SGP_{bool} indicates the use of switching generator-pointer.

5.4.2 DUC Corpus

The DUC corpus lacks the amount of data needed for training and was thus used for evaluation of various models in comparison to Rush et al.'s model ABS and ABS+ (Rush et al., 2015) and the best performing model on DUC 2004, TOPIARY (Zajic et al., 2004). Rush et al.'s models were based on a convolutional encoder instead of being RNN based. ABS+ was trained on the Gigaword Corpus (ABS) and tuned on the DUC-2003 validation set. In contrast, the model proposed by Nallapati et al. (2016), words-lvt2k-1sent (see Table 2) was trained only on the Gigaword corpus with no further tuning. It still performs consistently better than ABS+ both during Gigaword evaluation and DUC evaluation. Limited ROUGE Recall is used as an evaluation criteria for these models (Table 3(b))

5.4.3 CNN/Daily Mail Corpus

Both the Gigaword and DUC corpora have one sentence summaries. In contrast, the modified CNN/Daily Mail dataset proposed by Nallapati et al. (2016), contains multisentencial summaries.

Interestingly, the most basic model *words-lvt2k* yields higher performance statistics over the more sophisticated models containing hierarchical attention (*words-lvt2k-hieratt*) or switching generator-pointer

Model	Rouge-1	Rouge-2	Rouge-L	Src.copy Rate (%)
Full length F1 on internal test set of 2000 pairs				
words-lvt2k-1sent	34.97	17.17	32.70	75.85
words-lvt2k-2sent	53.73	17.38	33.25	79.54
words-lvt2k-2sent-hieratt	36.05	18.17	33.52	78.52
feats-lvt2k-2sent	35.90	17.57	33.38	78.92
feats-lvt2k-2sent-ptr	**36.40**	**17.77**	**33.71**	**78.70**
Full length Recall on the test set used by Rush et al. (2015)				
ABS+ (Rush et al., 2015)	31.47	12.73	28.54	91.50
words-lvt2k-1sent	**34.19**	**16.29**	**32.13**	**74.57**
Full length F1 on the test set used by Rush et al. (2015)				
ABS+ (Rush et al., 2015)	29.78	11.89	26.97	91.50
words-lvt2k-1sent	**32.67**	**15.59**	**30.64**	**74.57**

Model	Rouge-1	Rouge-2	Rouge-L
TOPIARY	25.12	6.46	20.12
ABS	26.55	7.06	22.05
ABS+	28.18	8.49	23.81
feats-lvt2k-1sent	**28.35**	**9.46**	**24.59**

Model	Rouge-1	Rouge-2	Rouge-L
words-lvt2k t	**32.49**	**11.84**	**29.47**
words-lvt2k-ptr	32.12	11.72	29.16
words-lvt2k-hieratt	31.78	11.56	28.73

Table 3: Evaluation of various models on the (a) Gigaword corpus, (b) DUC corpus and the (c) Dailymail/ CNN corpus. The values in these tables is taken from Nallapati et al. (2016). The bold values indicate the highest performance among all the models for a certain evaluation criteria. ABS+ is the convolutional encoder model by Rush et al. (2015), that is trained on Gigaword corpus and tuned on the DUC validation set.

mechanism (*words-lvt2k-ptr*). The authors conclude that this dataset needs further investigation and more experiments in order to draw any inferences from this observation.

5.5 Qualitative Evaluation

While, the system was noted to produce good quality summaries most of the times, it may misinterpret more complex sentences and produces summaries that diverge from the meaning of the input text. An example of a poor summary is provided by the authors (Nallapati et al., 2016):

- **Source**: broccoli and broccoli sprouts contain a chemical that kills the bacteria responsible for most stomach cancer , say researchers , confirming the dietary advice that moms have been handing out for years . in laboratory tests the chemical , $< UNK >$, killed helicobacter pylori , a bacteria that causes stomach ulcers and often fatal stomach cancers.
- **Reference Summary**: for release at $\#\#\#\# < UNK >$ mom was right broccoli is good for you say cancer researchers
- **System Summary**: broccoli sprouts contain deadly bacteria

6 Conclusion

In this work, sequential modelling tasks especially in the context of NLP are studied in detail. Focus is placed on Deep Learning approaches that are more generalized than rule-based systems and consume lower memory than statistical methods. We further learn, that encoder-decoder models form the basis of deep learning sequential modelling and reason why RNN's make a good choice for the encoder and the decoder. The limitations of encoder-decoder models with a fixed sized context vector are explored and the role of attention in combating this issue is elaborated upon. The role of hierarchical attention in interpreting multi-sentencial inputs is also explored, both in theory, as well as through statistical evidence produced by the application paper (Nallapati et al., 2016). Finally, the paper also produces results supporting the inclusion of feature rich inputs for a better performing system.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*.

Denny Britz. 2015. Recurrent neural networks tutorial. Http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 137–144. Coling 2008 Organizing Committee.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with lstm.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Conference of the Association for Machine Translation in the Americas*, pages 115–124. Springer.

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

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

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

Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence - video to text. *CoRR*, abs/1505.00487.

David Zajic, Bonnie Dorr, and Richard Schwartz. 2004. Bbn/umd at duc-2004: Topiary. In *Proceedings of the HLT-NAACL 2004 Document Understanding Workshop, Boston*, pages 112–119.