NLP Summarization: Abstractive Neural Headline Generation Over A News Articles Corpus

Conference Paper · October 2020

DOI: 10.1109/ICDS50568.2020.9268776

4 authors:

Mehdi Erraki
Université de Franche-Comté
2 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Youssfi Mohamed
Université Hassan II de Casablanca
191 PUBLICATIONS **927** CITATIONS
SEE PROFILE

Daaif Abdelaziz
Université Hassan II de Casablanca
16 PUBLICATIONS **33** CITATIONS
SEE PROFILE

Omar Bouattane
Université Hassan II de Casablanca
281 PUBLICATIONS **1,363** CITATIONS
SEE PROFILE

NLP Summarization : Abstractive Neural Headline Generation Over A News Articles Corpus

Mehdi Erraki
SSDIA Laboratory
ENSET Mohammedia
University Hassan 2
Casablanca, Morocco
mehdi.erraki@gmail.com

Mohamed Youssfi
SSDIA Laboratory
ENSET Mohammedia
University Hassan 2
Casablanca, Morocco
med@youssfi.net

Abdelaziz Daaif
SSDIA Laboratory
ENSET Mohammedia
University Hassan 2
Casablanca, Morocco
aziz@daaif.net

Omar Bouattane
SSDIA Laboratory
ENSET Mohammedia
University Hassan 2
Casablanca, Morocco
o.bouattane@gmail.com

***Abstract*—Most of NLP research fields (Translation, Classification, Dialogue Systems …) have been revolutionized by the rise of deep learning methods, which rely on the new dense and low-dimensional feature representation. We present in this article the basic training techniques of Word Embeddings as well as the recent works on Abstractive Neural Summarizers. We also introduce our trained French Word Embeddings, further used as the embedding layer to implement our baseline French Neural Summarizer for the headline generation task, using the RNN (Recurrent Neural Network) Encoder-Decoder architecture.**

Keywords— word embeddings, summarization, headline generation, recurrent neural networks, seq2seq, encoder-decoder, attention, pointer, reinforcement learning, big data

I. Introduction

Text summarization is an active field in NLP which consists in automatically generating an informative condensed natural language summary from a large input text. Summarizer systems are mainly used to help other downstream applications like search and multi-document reports generation.

The extractive methods, which are based on copying some raw parts from input text, have historically been widely and successfully used. The abstractive methods, consisting in generating summaries which may not contain original tokens and phrases, have always been a big challenge for summarization research, and have mainly been associated with hand-crafted syntactic and linguistic transformations. Since the majority of the latter systems are specific to the English language, a linguistic expertise is needed to transpose these works to other languages.

On the other hand, fully data-driven abstractive summarizers have emerged in the last years thanks to the rise of deep learning methods and the dense feature representation: Word Embeddings. The Seq2Seq model based on an RNN (Recurrent Neural Network) Encoder-Decoder architecture has first been introduced for the translation task, before inspiring the most recent English neural abstractive summarizers.

The challenge of our work is to use these techniques for French summarization without any language-specific adaptation. To achieve our work, we have first used web-scrapping to generate our own French news corpus. Then, using our homemade French Word Embedding, we have applied the state of art RNN auto-encoder techniques for the headline generation task and have compared our results to similar English models.

In this paper, we first introduce in section II the new distributed feature representation, and present our French Word Embeddings trained on a news articles corpus of 317 M token.

Next, in section III, we dive into recent works on neural summarizers based on the Seq2Seq architecture, as well as the additional adjustments to this model in order to enhance accuracy and solve repetition and long input text issues.

Finally, we introduce in section IV our baseline French Attention-Based Seq2Seq model for the headline generation task trained on news articles. We also describe its limits and introduce our future works in section V.

II. Distrubuted Feature Representation

A. Related Works

1) From Sparse to Dense Representation

Traditional shallow models (Naïve Bayes, Logistic Regression, SVM) have been massively used for decades on various NLP tasks [1]. These models are mainly trained on a high-dimensional space (d~50000), where the very sparse feature representation (often using one-hot vectors) suffers from the curse of dimensionality [2]. Distance between two elements is almost the same when using sparse features, and thus this representation is unable to capture syntactical and semantical similarities, and is often associated with a time-consuming and hand-crafted features extraction.

On the other hand, new deep learning methods based on a dense and low-dimensional feature representation (Word Embeddings) have recently proved to produce superior results on most of NLP tasks [1]. Also referred as the "Distributed Representation", Word Embeddings are based on the distributional hypothesis: words which occur in similar contexts share similar meanings.

Typically, Word Embeddings are used as the first data layer to encode input words as dense input vectors, using an embedding lookup table. They are used in "end-to-end" deep learning models trained on huge amounts of data, enabling automatic feature representation. The rows of the lookup table (i.e. word vectors), which are trained along with other model's parameters, could be initialized either randomly or based on pre-trained vectors. Models like "Word2vec" [3], "GloVe" [4], and "FastText" [5] could be used to "pre-train" word vectors.

2) Word2Vec Models

The success of Word Embeddings is attributed to the introduction of the "Word2vec" algorithm [3] in its two variants : CBOW and Skip-Gram models. Trained on a very

978-1-7281-8084-7/20/$31.00 ©2020 IEEE

large and unlabeled corpus (e.g. Wikipedia corpus) as a language model [6], Word Embeddings are then extracted from the model's weights.

Basically, the Word2vec models are based on a 3-layer feed-forward neural network (Fig. 1). The input layer is V-dimensional taking one or the sum of a few one-hot word vectors (V~100000 : vocabulary's size). The hidden-layer (projection) is a linear d-dimensional layer (d~100). The output layer is a V-dimensional SoftMax over the vocabulary, technically implemented using Hierarchical SoftMax [3] or Negative Sampling [7].

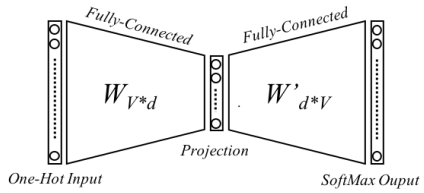

Fig. 1. CBOW and Skip-Gram arcitechture

Given a window size k, each [2*k+1]-windowed sequence of a document from a large corpus is used to produce :

- **CBOW** : one training example where the input sum of the 2*k one-hot context words vectors is projected into the low-dimensional space, before being passed to the SoftMax layer to predict the central word.
- **Skip-Gram** : 2*k training examples where the input one-hot vector corresponding to the central word is, at each time, projected to the low-dimensional space, and then passed to the SoftMax layer to predict one of the 2*k context words.

The learned projection matrix W_{V*d} corresponds to the extracted Word Embeddings, which could be used to fix or to initialize the embedding layer (embedding lookup table) of some neural network model for a given NLP task.

For our French neural headline generator trained on news articles (section IV), we will be using our French Word Embeddings, trained on news articles as well, so as to better capture relations within topic-varied media and actuality vocabulary.

B. *Training French Word Vectors*

We have built French Word Embeddings over a web-scrapped corpus of news articles using *Gensim*'s Word2Vec library.

1) Corpus

Over 750000 French news articles from diverse topics ranging from 90's to 2019 have been tokenized and parsed using the *TreeTagger* library. The created corpus contains 317 M tokens. Except lowercasing, tagging named entities with "$nam_" prefix and numbers with "$nb_" prefix, no further preprocessing has been applied.

2) Model

We trained a Skip-Gram model with Negative Sampling using the following settings:

TABLE I. SKIP-GRAM PARAMETERS

	Parameters		
Skip-Gram model	Vocabulary's size	V	300415
	Word vectors size	d	50
	Window's size	k	5
	Negative sampling	ns	5

Given an input training sequence $w_1, w_2, \ldots, w_T$, the objective function to maximize using Skip-Gram is :

$$1/T \sum_{i=1}^{T} \sum_{\substack{-k \le j \le k \\ j \neq 0 \\ 0 \le i+j \le T}} log\ p(w_{i+j} \mid w_i) \qquad (1)$$

While the hidden layer only projects sparse input, the SoftMax is a bottlenecked layer whose computing cost is proportional to V. Negative Sampling is used to replace each log of SoftMax probability term in (1) into a sum of logs of a binary probability (Logistic Regression) over ns + 1 words : the base-truth "positive" word, and ns "negative" words sampled from vocabulary.

3) Similarity and Analogy

The cosine measure could be used to evaluate similarity between words.

Using PCA (Principal Components Analysis) to reduce word vectors space's dimension from d=50 to d'=2, the scatter plot (Fig. 2), drawn from a set of selected words, helps visualizing topic-varied clusters of semantically related words.

One interesting characteristic captured by Word

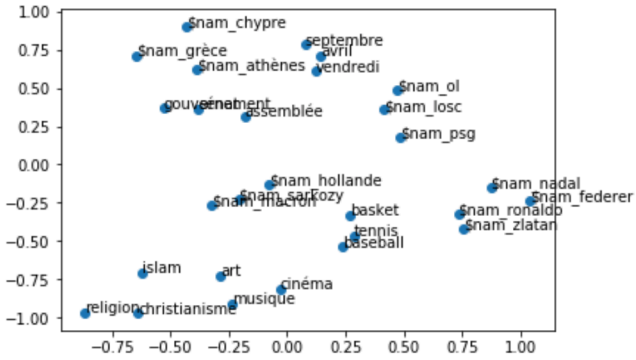

Fig. 2. 2-dimension scatter plot of a set of reduced word vectors

Embeddings is the analogy pattern as in: "Paris is for France what Rome is for Italy". Typically, we would like to have the word vector w_{France} as the most similar one to: $w_{Italy} - w_{Rome} + w_{Paris}$. Table II shows the three most similar words for some topic-varied analogy tests.

III. NEURAL TEXT SUMMARIZATION : RELATED WORKS

In this section, we give an overview of recent works on neural summarizers. Text Summarization in NLP is the task of automatically condensing information from an input text document into informative and short text summaries.

TABLE II. ANALOGY TESTS

Analogy Tests	Three most similar words	
	word	*cosine*
W$nam_espagne - W$nam_madrid + W$nam_londres	W$nam_grande-bretagne W$nam_singapoure W$nam_ inde	**0.85** 0.83 0.82
Wump - W$nam_sarkozy + W$nam_pen	Wfn Wfrontiste W$nam_front	**0.88** 0.86 0.82
Wnba - W$nam_basket + W$nam_football	Wnfl W$nam_mlb W$nam_nhl	**0.81** 0.79 0.76
Wprince - Whomme + Wfemme	Wprincesse Wduchesse W$nam_diana	**0.89** 0.85 0.83
Wcinéma − Wfilm + Wchanson	Wmusique Wpop Wmusicale	**0.86** 0.80 0.78
Wrap − W$nam_eminem + W$nam_marley	Wreggae Wbreackdance Wrock	**0.78** 0.76 0.75

Summarization algorithms are classed into two main categories:

- **Extractive summarization** which is based on copying some of the most informative raw parts as they are from the input text.
- **Abstractive summarization** which produces a text sequence that may contain words and paraphrases which are not present in the original text.

We only focus on abstractive summarization which has gained recently much more importance thanks to the rise of the RNN and the Word Embeddings.

A. Seq2Seq Model

First introduced for the machine translation task [8], the Seq2seq model, based on an RNN Encoder-Decoder architecture, has been massively used for the summarization task [9] [10] [11] [12].

We define the input (*resp.* output) word vectors sequence as $w_x^1, w_x^2, ..., w_x^{Tx}$ (*resp.* $w_y^1, w_y^2, ..., w_y^{Ty}$). The vocabulary size is referred as V, and the RNN internal transformation as *rnn* ().

The basic model (Fig. 3) contains two components:

- **The encoder**: uses an RNN to encode the input sequence into a fixed-length context vector c_x. In [8], c_x is simply equal to the last hidden state of the RNN.

$$h_x^k = rnn_x\,(h_x^{k-1}, w_x^k) \tag{2}$$

$$c_x = h_x^{Tx} \tag{3}$$

- **The decoder**: uses an RNN and is trained to predict at each time step t the ouput word w_y^t based on the context vector c_x and the current hidden state of the RNN h_y^t. The latter relies on the previous word (predicted at test time w_y^{t-1}, or teacher forced at train time w_y^{t-1}*).

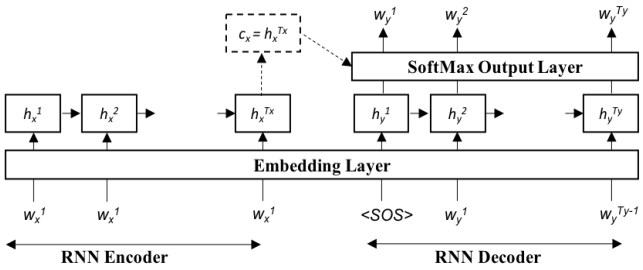

Fig. 3. Arcitechture of the Seq2Seq Encoder-Decoder

$$h_y^t = rnn_y\,(h_y^{t-1}, w_y^{t-1}) \tag{4}$$

$$P_{out}\,(w_y^t) = softmax\,(W_{out}\,[h_y^t \,||\, c_x] + b_{out}) \tag{5}$$

The loss function is the average of all decoding steps cross-entropy losses :

$$loss = 1/T_y \sum_{t=1}^{Ty} loss^t \tag{6}$$

$$loss^t = -\log\,(P_{out}(\,w_y^t\,*)) \tag{7}$$

While this model has proved to produce good results when evaluated using ROUGE metrics (measures n-gram overlap between generated and ground-truth sequences), it still has some limits.

First, it tends to produce unnatural summaries which suffers from repetition of words and phrases. Also, the fixed-size vector is unable to encode all information from long sequences. And finally, this model isn't able to output Out Of Vocabulary (OOV) words, which may be crucial for summarization.

In next sub-sections, we introduce the adjustments proposed to deal with the three latest issues.

B. Attention Over Input

First proposed for the translation task [13], the mechanism of attention (Fig. 4) has also been adopted for the Seq2Seq summarizer [9], allowing to encode information from long input sequences (50-60 words).

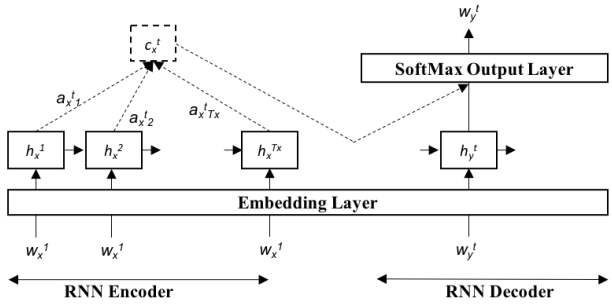

Fig. 4. Attention over input at each time decoding step t

The main difference from the basic Seq2Seq model is that the decoder's prediction is not based anymore on a single fixed-length vector c_x at all decoding steps. It relies, instead, at each decoding time step t, on an a weighted distribution

over all encoder's hidden states $h_x^1,\ldots, h_x^{Tx}$ with respect to the current decoding hidden state h_y^t :

$$P_{out}(w_y^t) = softmax(W_{out}[h_y^t \,||\, c_x^t] + b_{out}) \quad (8)$$

$$c_x^t = \sum_{k=1}^{Tx}(a_x^t{}_k \,.\, h_x^k) \quad (9)$$

$$a_x^t{}_k = softmax^{k\ in\ [1:Tx]}(s_x(h_y^t, h_x^k)) \quad (10)$$

where $s_x()$ is a scoring function varying from dot-product (11), to a linear feedforward neural network (12); whose weights W_{att} and b_{att} are learned jointly with other model's parameters.

$$s_x(h_y^t, h_x^k) = {}^{tr} h_y^t \,.\, h_x^k \quad (11)$$

$$s_x(h_y^t, h_x^k) = W_{att}[h_y^t \,||\, h_x^k] + b_{att} \quad (12)$$

This function measures the alignment between the current decoding hidden state h_y^t and a given encoding hidden state h_x^k, thus representing how much attention to give to word w_x^k in order to predict word w_y^t.

C. Intra-Decoder Attention and Coverage Mechanism

The two following methods have been proposed in order to solve the repetition issue.

In [12], the intra-decoder attention is introduced as an attention mechanism over previous decoding steps. Initialized to zero, at each decoding time step t, a decoding context vector c_y^t is computed over previous decoding hidden states $h_y^1, \ldots, h_y^{t-1}$. It uses an alignment function $s_y()$ which is similar to $s_x()$.

$$c_y^t = \sum_{k=1}^{t-1}(a_y^t{}_k \,.\, h_y^k) \quad (13)$$

$$a_y^t{}_k = softmax^{k\ in\ [1:t-1]}(s_y(h_y^t, h_y^k)) \quad (14)$$

The intra-decoder attention c_y^t is then fed to the output layer, along with the temporal attention c_x^t, and the current hidden state h_y^t; in order to additionally learn to generate better sequence structure and prevent repetition.

$$P_{out}(w_y^t) = softmax(W_{out}[h_y^t \,||\, c_x^t \,||\, c_y^t] + b_{out}) \quad (15)$$

The second method, introduced in [11] as the coverage mechanism, modifies the temporal attention c_x^t through including a coverage vector cov^t in the attention scoring function in (10) :

$$a_x^t{}_k = softmax^{k\ in\ [1:Tx]}(s_x(h_y^t, h_x^k, cov^t{}_k)) \quad (16)$$

$$s_x(h_y^t, h_x^k, cov^t{}_k) = W_{att}[h_y^t \,||\, h_x^k \,||\, cov^t{}_k] + b_{att} \quad (17)$$

where cov^t is the sum of all previous input attention distributions :

$$cov^t = \sum_{t'=1}^{t-1}(a_x^{t'}) \quad (18)$$

The vector cov^t represents the cumulated input attention distribution so far until last decoding step.

Also, a coverage loss $covLoss^t$ is included in the loss function in (7) with a certain weight α, in order to penalize the input attention mechanism when it attends the same input location in different decoding steps :

$$loss^t = -\log(P_{out}(w_y^t *)) + \alpha\, covLoss^t \quad (19)$$

$$covLoss^t = \sum_{k=1}^{Tx}(min(c_x^t{}_k, cov^t{}_k)) \quad (20)$$

The coverage loss is a bounded value (the sum of $c_x^t{}_k$ is equal to 1), attaining the biggest value when the current attention distribution c_x^t at decoding time t is similar to the cumulated attention coverage cov^t.

D. The Pointer Mechanism

So far, the output layer in (5), (8), and (15) generates a SoftMax distribution $P_{out}(w_y^t)$ over the V words of the vocabulary. One limit of this model is its inability to generate OOV words which are the rare or unseen words, that might be important for the summarization task though.

The pointer mechanism has been introduced in [10] for the Seq2Seq summarizer in order to solve this issue. At each decoding time, the decoder uses a switch function as the probability of either using the SoftMax output generator (s=1), or using a pointer to extract a word from the input document sequence (s=0). The switching function corresponds to a one-layer feed-forward neural network with a sigmoid function taking as input the current decoding hidden state h_y^t, the input context vector c_x^t, and the last emitted word w_y^{t-1}.

$$P(s=1) = \sigma(W_{switch}[h_y^t \,||\, c_x^t \,||\, w_y^{t-1}] + b_{switch}) \quad (21)$$

When s=0, the pointer mechanism is applied to copy an input word; sampled using the input attention distribution.

In [11], a softer pointer mechanism is proposed instead of the "hard" switch, in order to introduce the input attention distribution within the output distribution:

$$P(w_y^t) = P(s=1)P_{\text{out}}(w_y^t) + P(s=0)\sum_{k}^{Wyt=Wxk}(c_x^t{}_k) \quad (22)$$

IV. Baseline Attention-Based Seq2Seq Model for French Headline Generation

The headline generation task when applied to news articles could be assimilated to the summarization one with the particularity of having a very condensed ouput: one sentence corresponding to the document's headline, and while the input document may be very long.

Some approaches propose a coarse-to-fine method [14] to adapt the Seq2Seq summarizer to the headline generation task, and others [9] propose truncating the input documents to smaller text (ranging from the first sentence to some hundreds of tokens).

Using a corpus of web scrapped French news articles, we followed the truncating approach to train an Attention-Based Seq2seq model which will be used as a baseline in our future works.

A. Corpus :

The corpus corresponds to 239526 French news articles which are labelled (i.e. each article document is associated with its headline).

A truncation to a maximum of 75 tokens (*resp*. 25 tokens) has been applied to input articles (*resp*. output headlines). Training samples having input text with less than 15 tokens or headlines with less than 3 tokens have been filtered out. The model's vocabulary is composed of the V-most frequent words

(V = 30000), and a few reserved words (like "$unk" used to replace O.O.V words).

B. Model and Training

The trained model corresponds to the Attention-Based Seq2Seq (ABS), as presented in Fig. 3 and described by (8), (9), (10), and (12). Our trained French Word Vectors introduced in section II are used to initialize the embedding layer, which will be further trained.

In both the encoder and the decoder, the *rnn* () function is implemented using a GRU (Gated Recurrent Unit) with a single 150-sized hidden layer. Similarly to [9], we have implemented a bi-directional encoder, which corresponds to two GRU networks reading the input sequence in both directions. The encoder hidden state h_x^k corresponds to the concatenation of their hidden states $[h_x^k{}_{forward} \,||\, h_x^k{}_{backward}]$.

Training, validation, and testing are done respectively over 85%, 5%, and 10% of data. The whole model architecture is implemented using "*PyTorch*". One epoch takes about 8 hours to complete on a single CPU. We also used partial and decaying teacher forcing. Early stopping is applied to keep the best evaluated model on validation set using ROUGE metric.

C. Results

In table III, we report two evaluations of our model on the test data using the ROUGE score.

TABLE III. MODEL EVALUATION

Model	ROUGE SCORE		
	R-1	*R-2*	*R-SU4*
F-(ABS)-75 (0.2 M articles)	20.03	6.67	7.75
F-(ABS) (0.2 M articles)	18.03	5.9	6.86
E-(ABS) (4 M articles)	**26.55**	**7.06**	-

In F-(ABS)-75, test input documents are truncated to a maximum of 75 tokens as in training, while in F-(ABS) no truncation is applied. We also indicate as a reference the ROUGE score of the English Attention-Based Summarization system E-(ABS) from [9], trained on the "Gigaword" corpus [15].

While these results are encouraging regarding the relatively low volume of data used for training, our baseline model still faces some limits.

First, we noticed performance losing once it is scaled to longer input document size.

We also observed on some specific examples that some key information (e.g. the value of Paris stock market's rise) would better be trained to be retrieved as-is from input sequence, while our model tends to generate arbitrary values.

Finally, it generally tends to generate repeated tokens over decoding times:

- [**baseline**] *"$nam_netbooster : augmentation du nombre d' actions en circulation"*
- [**predicted**] *"$nam_netbooster : augmentation du nombre d' actions en circulation"*
- [**baseline**] *"$nam_afghanistan : attentat contre un convoi de l' otan à $nam_kaboul , un civil tué"*
- [**predicted**] *"$nam_afghanistan : un morts dans un attentat"*
- [**baseline**] *"$nam_syrie : les $nam_kurdes repoussent une incursion des jihadistes à $nam_kobané"*
- [**predicted**] *"$nam_syrie : les jihadistes tués à à à $nam_kobané $nam_kobané"*
- [**baseline**] *"$nam_wall $nam_street finit en hausse , encouragée par la $nam_chine et les résultats"*
- [**predicted**] *"$nam_wall $nam_street finit en hausse , la la"*
- [**baseline**] *"la bourse de $nam_paris au plus haut depuis janvier $nb_2008"*
- [**predicted**] *"la bourse de $nam_paris finit en hausse de % %"*
- [**baseline**] *"$nam_valneva : succès de l' augmentation de capital"*
- [**predicted**] *"$nam_valneva : augmentation d' augmentation capital capital capital"*
- [**baseline**] *"orages , inondations : $nb_21 départements placés en vigilance orange"*
- [**predicted**] *"orages : les départements dans orange orange orange orange"*

V. CONCLUSION AND FUTURE WORKS

We presented in this paper the recent trends in RNN-based neural summarizers enhanced by the rise of the distrusted feature representation.

We introduced our French Word Embeddings, used to initialize the embedding layer of our baseline French Attention-Based Seq2Seq model for headline generation. We showed the limits of this model, which have also been introduced in section III, along with the state-of-art adjustments helping to solve them.

In our future works, we will improve training capacities and volumes, as well as the model's architecture and objective.

A. Training Using GPU and a Big Data Framework

We are preparing a French news articles corpus of 1.3 M articles for the headline generation task. To leverage this volume, models will be trained on GPU.

We would also aim at using the Big Data Spark framework for fine-tuning, through parallelizing separate models training. Only using the mapping function (i.e. no shuffling transformations), a cluster of N of k-CPU (or k-GPU) workers accessing locally stored data would allow to train $N*k$ models simultaneously.

B. Solving Observed Issues

To deal with the repetition issue, both the intra-decoder attention and the coverage mechanism will be added to the baseline model for evaluation.

The pointer mechanism will also be implemented to handle OOV words, and to enable key-information extraction from input.

Input attention helps encoding information from long input sequences (up to 50 tokens) [9]. For longer input documents, we would proceed by augmenting the RNN complexity and investigating hierarchical encoding [14]. The

latter might help to adapt our summarizer model to the particularity of the headline generation task (i.e. encoding long text to generate very short text).

C. *Reinforcment Learning*

The ROUGE metric is non-differentiable and while it is used to evaluate our summarizer model, it is not used in the training objective.

The Seq2Seq model is based on a language model's maximum likelihood objective, penalizing inaccurate word generation at each decoding time step. It is thus less adapted to the summarization task, where there is potentially a large number of valid summaries.

Metrics measuring n-gram overlap like the ROUGE and the BLEU scores are more flexible to the sequence's order and structure than the language model objective. Our next works will aim at introducing a reinforcement self-critical policy inspired from [12], so that the reward expectation of the model is still important when the ouput distribution for sequence generating allows obtaining high ROUGE value despite of a relatively high language model loss.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grand agreement No 777720.

REFERENCES

[1] T. Young, D. Hazarika, E. Cambria, and S. Poria, "Recent Trends in Deep Learning Based Natural Language Processing", IEEE Computational Intelligence Magazine, Vol 3, Issue 3, August 2018 pp. 55 – 75.

[2] R. Bellman, "Dynamic programming", Princeton, NJ: Princeton University Press, 1957.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space", Proceedings of the International Conference on Learning Representations (ICLR 2013), January 2013, pp. 1-12.

[4] J. Pennington, R. Socher, C.D. Manning, "GloVe: Global Vectors for Word Representation", Conf Proc 2014 Conf Empir Methods Nat Lang Process, 2014, Vol 14, pp. 1532-1543.

[5] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov , "Enriching Word Vectors With Subword Information", Transactions of the Association for Computational Linguistics, July 2016.

[6] Y. Bengio, R. Ducharme, P. Vincent , C. Janvin, "A neural probabilistic language model", The Journal of Machine Learning Research, Vol 3, Janvier 2003, pp. 1137-1155.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality", NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems, Vol 2, December 2013, pp. 3111–3119.

[8] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks", Advances in neural information processing systems, 2014, pp. 3104-3112.

[9] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization", Proceedings of EMNLP, 2015.

[10] R. Nallapati, B. Zhou , C.Gulcehre, B. Xiang, et al., "Abstractive text summarization using sequence-to-sequence rnns and beyond", Proceedings of SIGNLL Conference on Computa- tional Natural Language Learning, 2016.

[11] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks", Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, pp. 1073-1083.

[12] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization", ICLR 2018.

[13] D. Bahdanau, K. Cho, and Y.Bengio., "Neural machine translation by jointly learning to align and translate", Proceedings of the International Conference on Learning Representations, 2015.

[14] J. Tan, X. Wan, J. Xiao , "From neural sentence summarization to headline generation: a coarse-to-fine approach", IJCAI'17 Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 4109-4115.

[15] D.Graff, C. Cieri, "English Gigaword", Linguistic Data Consortium (LDC), catalog number LDC2003T05, ISBN 1-58563-260-0, January 2003.