Implementation of Extractive Document Summarization Based on Convolutional Neural Networks - Final Milestone

Leo Laugier - 3033157038
Master's Student
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
leo_laugier@berkeley.edu

Evan Thompson - 3033157454
Master's Student
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
thompe5@berkeley.edu

Alexandros Vlissidis - 3033156349
Master's Student
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
alex_vlissidis@berkeley.edu

I. Problem Definition and Motivation

Extractive Summarization is a method, which aims to automatically generate summaries of text. This task is challenging because compared to key-phrase extraction, text summarization needs to generate a whole sentence that described the given document, instead of just single words. Another challenge for this task has been the manual generation of text summaries for supervised learning. Legacy algorithms use predefined handcrafted features of the text for representation. This makes it extremely painful to build an effective algorithm. We are implementing an algorithm described by Y. Zhang et al.[1], where a Convolutional Neural Network (CNN) [2] approach is proposed. This has the benefit that it can use word-embedding to represent text and the neural network can extract features automatically.

CNNs have traditionally been used in computer vision tasks, since this was their intended original use [3]. However, recent work has shown that they are very effective in NLP [4] tasks as well. The specific model we are implementing is a regression process for sentence ranking. The architecture of this method consists of a convolution layer followed by a max-pooling layer, on top of a pre-trained *word2vec* [5],[6] mapping. We will implement this new proposed method and perform experiments on single and multi-document summarization. The original paper purports that it can achieve better than state-of-the-art systems. Our goal is to verify the results and extend the analysis using more extensive evaluations, on different datasets and metrics. In addition, we will open source our implementation. This is important because this method does not require any prior knowledge in the text summarization field, hence can be used by anyone and provide state-of-the-art performance.

II. Related Work and Comparisons

During the last few years there have been many publications focused on using deep learning methods for text summarization documents. There are two types of document summarization; extractive and abstractive. The former uses sentences from the given document to construct a summary (and is considered an easier task), and the latter generates a novel sequence of words using likelihood maximization. As mentioned in the introduction we are focusing on related work in extractive text summarization. SummaRuNNer [13] achieves comparable to state-of-the-art performance in single document text summarization. It proposes an GRU-RNN network, which gives the advantage of having a model that is easily interpretable. However, this method only in some cases achieves state-of-the-art performance and does not propose a method to extend to multi-document summarization. RNNs have been used to rank sentences in [14], showing incredible performance in this task. This proposed method is limited by the fact that it uses hand-crafted features to represent the input to the RNN. Probably the closest paper to the one we are implementing is [15], which makes use of a hierarchical CNN to summarize documents. The advantage of the method we are implementing is that it has a much simpler architecture and is therefore more computationally inexpensive and is easier to tune. Most recently a Restricted Boltzmann Machine [16] (RBM), is proposed in [17]. However, it is only applied to extractive document summarization in factual reports.

III. Approach

In our implementation we decided to use Python, due to its versatility and fast production capabilities, as well as the great support it has from deep learning frameworks. We also decided to implement this project in TensorFlow, since it allows us low level access of deep learning constructs for experimentation, while maintaining ease of use. In addition, we are going to use ROUGE-1 and ROUGE-2 [4] as an evaluation metric in order to be able to compare our model's performance with the original paper and the state-of-the-art systems. After the project proposal phase, the code implementation was split into three main stages.

A. Data Pre-processing

Before the CNN training process, pre-processing stages are needed on the *DUC*[1] dataset. First of all, the DUC documents and summaries are stored in XML file format. The design of the data loader module needs to be efficient as well as versatile, as in the early stages of the project it is likely that design changes will be made.

The next stage of our pre-processing pipeline will need to split up the documents into sentences and obtain their word embedding using Google's pre-trained *word2vec*[2], which was trained on 100 billion words from the Google News dataset leading to an embedding of 3 million words at $k = 300$ latent dimensions. At the same time, the salience score for each sentence can be computed using the *pyrouge*[3] Python module. The salience scores for each sentence, as well as its word embedding, will be stored in pickle file formats for multiple reasons; it allows fast loading during training, but also it enables us to easily change our pre-processing stages without affecting the training stage.

B. Model

The model we used was described by [8] and is composed of one convolution layer, one max pooling , one dropout layer and one fully connected layer.

1) Input: $\forall i$, we represent the i^{th} sentence as a $n \times k$ matrix where $\forall j \in [\![1; n]\!]$ each row $x_{i,j}^\top$ is the k-dimensional representation of the j^{th} word of the i^{th} sentence. We will pad it correctly so that every sentence has the same standard length $n \in \mathbb{N}$.

$$x_i = \begin{bmatrix} x_{i,1}^\top \\ x_{i,2}^\top \\ \vdots \\ x_{i,n}^\top \end{bmatrix}$$

We will have an interesting design choice here: we will compare the *word2vec* initialization of x_i with a random initialization. The random initialization will involve an Embedding layer which weights will be updated at each Stochastic Gradient Descent iteration.

2) 1D Convolutional layer: In order to extract the sentences' meaning, we begin our network with l one-dimension convolutions filters $(w_f)_{f \in [\![1;l]\!]} \in \mathbb{R}^{k \times m}$; $m \in \mathbb{N}$ being the window size of the filters.

$\forall (f, j) \in [\![1; l]\!] \times [\![1; n-m+1]\!]$, we define a feature $c_{i,f,j} = \sigma_1(\langle w_f, \left[x_{i,j}, x_{i,j+1}, \ldots, x_{i,j+m-1}\right] \rangle_F + b_{f,j})$ where $b_{f,j} \in \mathbb{R}$ is the bias term and σ_1 is the first activation function such as ReLu, Sigmoid or Hyperbolic Tangent ;

$c_{i,f} = \begin{bmatrix} c_{i,f,1} \\ c_{i,f,2} \\ \vdots \\ c_{i,f,n-m+1} \end{bmatrix} \in \mathbb{R}^{n-m+1}$ is called a feature map.

Therefore, for each feature map $f \in [\![1; l]\!]$ we build $n - m + 1$ features capturing the meaning of m adjacent words and produce one feature per feature map.

3) Max pooling layer: Then, $\forall f \in [\![1; l]\!]$, we keep $h_{i,f} = max((c_{i,f,j})_{j \in [\![1;n-m+1]\!]})$ through a max pooling layer so that we keep the important feature.

4) Dropout layer: The fully connected layer is prone to overfitting so we regularize the model thanks to a dropout layer.

Let $r = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_l \end{bmatrix} \in \mathbb{R}^f$ be the masking vector with probability p.

The input to the fully connected layer is $h_i \odot r$.

5) Fully connected layer: We eventually apply a Fully Connected layer with a sigmoid activation and a single output.
$\hat{y_i} = \sigma_2(w_r \cdot (h_i \odot r) + b_r) \in \mathbb{R}$ where $w_r \in \mathbb{R}^l$, $b_r \in \mathbb{R}$ and σ_2 is the sigmoid activation function.

6) Output: The output is a scalar $\hat{y_i} \in \mathbb{R}$ which we aim to approximate the salience score $y_i = \alpha \cdot R_{i,1} + (1-\alpha) \cdot R_{i,2} \in \mathbb{R}$ of the sentence x_i, by training the neural network. $R_{i,q}$ is the ROUGE-q score for the i^{th} sentence.

C. Training

For training, we will load embeddings for words from pickle files and load the saliency scores for all sentences.

1) Constraint: During training, we impose a l_2-norm constraint to w_r for regularization purpose.

2) Loss function: The loss function we minimize in training is the binary cross-entropy:

$$\mathcal{L} : \mathbb{R}^2 \longrightarrow \mathbb{R}$$
$$(y, \hat{y}) \longmapsto \mathcal{L}(y, \hat{y}) = -y \cdot ln(\hat{y}) - (1 - y) \cdot ln(1 - \hat{y})$$

3) Network parameters: We will train our network with Adadelta as Stochastic Gradient Descent update rule, used as an efficient backpropagation algorithm.

The network has a total of $l \cdot (m \cdot k + 1)$ weights: $l \cdot m \cdot k$ for $(w_f)_{f \in [\![1;l]\!]}$ and l for w_r.

It has $l \cdot (n - m + 1) + 1$ biases: $l \cdot (n - m + 1)$ for b and 1 for b_r.

[1] https://duc.nist.gov/

[2] https://code.google.com/p/word2vec

[3] https://pypi.python.org/pypi/pyrouge/0.1.0

D. Testing

For a new document, we input every sentence to the network so as to compute the sentences' saliency scores. We rank the sentences according to their saliency scores. As we want to build a summary with non-redundant sentences, we select sentences based on [9]. First we add the sentence with the highest score to the empty summary, then we repeat until the length limit of the final summary is met: select the sentence with the next highest score, if the similarity of the sentence with the summary being built is less than a threshold t, then add the sentence to the summary.

Furthermore we will need to cross validate to tune the hyperparameters n, l, m, p, α and t. We can also see how σ_1 and how changing the window size of some filters affect the results. We also consider testing on Single/Multi Document Summarization (ROUGE-1 and ROUGE-2) and compare our results to with state of the art especially Wpdv-xtr.v1 [10] and Cssnsa.v2 [11].

IV. Current Progress

Our progress is primarily confined to our detailing of the approach. We spent significant time attempting to fully understand the paper that our implementation is based on and creating a detailed outline of how we should approach the code. In addition to this development of the project outline, we also explored in detail the other technologies that we were less familiar with such as ROUGE. We found packages for dependencies and got our development environments appropriately setup to begin work on the project.

We have also setup our GitHub to keep track of our progress and tasks. We are creating our tasks on GitHub's Issues feature and using their milestone feature to make sure that are holding each other accountable.

The current iteration of the code is non-functional. However, there is a python-esque pseudo-code of our full algorithm and implementation that we allow us to keep track of what portions of the project fit together and need to be implemented.

V. Timeline

TABLE I: Timeline.

3/26	Spring Break
3/30	Model Implemented
4/2	Preprocessing task
4/09	Start training / Hyperparameter Tuning
4/16	Analyze results and investigate rouge-2 issues, Comparison with state-of-the-art as well as the paper we are implementing
4/23	Poster session
4/27	Final Results and Project Report Submitted

References

[1] Y. Zhang et al, *Extractive Document Summarization Based on Convolutional Neural Networks*, IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, p. 918-922, 2016.

[2] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[4] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[5] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. *NIPS 2013*.

[6] Chen, K., Corrado, G.S., Dean, J., Mikolov, T. (2013). Efficient Estimation of Word Representations in Vector Space. *CoRR, abs/1301.3781*.

[7] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 2004, pp. 74–81.

[8] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[9] Y. Li and S. Li, "Query-focused multi-document summarization: Combining a topic model with graph-based semi-supervised learning," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2014, p.11971207.

[10] H. van Halteren, "A Default First Order Family Weight Determination Procedure for WPDV Models" in *Proceedings o/CoNLL-2000 and LLL-2000*, pages 119-122, Lisbon, Portugal, 2000.

[11] J. M. Conroy, J. D. Schlesinger, D. P. O'Leary, M. E. Okurowski *Using HMM and Logistic Regression to Generate Extract Summaries for DUC* in Proceedings of the document understanding conference.

[12] P. Goyal, L. Behera, and T. M. McGinnity, "A context-based word indexing model for document summarization", *Knowledge and Data Engineering*, IEEE Transactions on, vol. 25, no. 8, pp. 1693–1705, 2013.

[13] R. Nallapati, F. Zhai and B. Zhou, "SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents", *The Thirty-First AAAI Conference on Artificial Intelligence*, 2016.

[14] Z. Cao, F. Wei, L. Dong, S. Li, and M. Zhou, "Ranking with recursive neural networks and its application to multi-document summarization", in *Proceedings of the 2015 AAAI Conference on Artificial Intelligence. AAAI*, 2015.

[15] M. Denil, A. Demiraj, N. Kalchbrenner, P. Blunsom, and N. de Freitas, "Modelling, visualising and summarising documents with a sin- gle convolutional neural network," *arXiv preprint arXiv:1406.3830*, 2014.

[16] Larochelle, H. Bengio, "Classification using discriminative Restricted Boltzmann Machines", *Proceedings of the 25th international conference on Machine learning - ICML*, p.536, 2008.

[17] S. Verma and V. Nidhi, "Extractive Summarization using Deep Learning", *arXiv:1708.04439v1*, 15 Aug. 2017.