Metamodel for ranking summarization models based on Doc2vec neural document representation and ROUGE scores
(SOP Seminar work)

Aleš Žagar

University of Ljubljana, Faculty of Computer and Information Science,
Večna pot 113, 1000 Ljubljana, Slovenia
ales.zagar@fri.uni-lj.si

1 Introduction

Text summarization is a task and research field within natural language processing that studies how to automatically produce short summaries from larger texts. It helps to reduce the time spent reading the whole document. In general, summarization models are divided into extractive, abstractive, and hybrid, based on the output they produce. Extractive models only copy essential parts of a text, whereas abstractive models generate new content which is more similar to how human abstracts look.

We produced many summarization models within the RSDO project[1]. The final 4 models belong to different approaches, using different methods and techniques, and have various pros and cons. Neural models work the best, but they do not generalize as well as unsupervised models. Neural models that were not trained on similar texts may produce summaries of worse quality compared to unsupervised or more simple approaches. This constitutes the problem of which model is the most appropriate when presented with a new text, since texts can be short, long, and of various genres, and they can come from almost anywhere when used in production.

Our solution[2] to this problem consists of a neural metamodel that automatically selects the most appropriate summarization model. We use a Doc2Vec neural representation of the input document and a simple two-layer fully connected neural network that predicts ROUGE scores for each summarization model. To get the final model, we combine these scores and select the most promising model for the document. Our solution belongs to the multi-target regression type of problem.

The rest of the seminar is divided into the following sections: We first describe how we built the dataset and which summarization models we used. After that, we present a method and results, and finally some weaknesses of our approach and ideas for how we can improve it.

[1] https://www.cjvt.si/rsdo/
[2] Code is available at https://github.com/azagsam/metamodel

2 Dataset

Our dataset consists of samples from various datasets. STA dataset (general news articles from Slovenian Press Agency) was extracted from Gigafida 2.0[3] and the first paragraph of each article was used as a proxy for summary since the dataset does not contain hand-written human summaries. This is a common technique in text summarization, especially in languages that do not have dedicated news article summarization datasets such as English. AutoSentiNews[4] is a similar dataset to STA, consisting of articles from the Slovenian news portals 24ur, Dnevnik, Finance, Rtvslo, and Žurnal24. The summaries are produced in the same way as in STA. SURS is a special financial news dataset from the Slovenian statistical office. The KAS corpus of Slovene academic writing consists of BSc/BA, MSc/MA, and PhD theses written from 2000 - 2018 and gathered from the digital libraries of Slovene higher education institutions via the Slovene Open Science portal (http://openscience.si/). The statistics for each dataset are enumerated in Table 1.

Dataset	**Number of documents**
STA	334,696
AutoSentiNews	256,567
SURS	4,073
KAS	82,308
Total	**677,644**

Table 1: Corpora and datasets used to create a final dataset.

We first concatenated all datasets. Next, in the preprocessing step, we removed high-frequency words that do not contribute to the meaning of a document, such as pronouns, conjunctions, etc., and to further reduce the number of different words, we lemmatized the whole dataset. This is an important step, especially when working with rich morphological languages such as Slovene. The full dataset was used to train the Doc2vec model.

For our metamodel, we randomly selected 93,419 examples from the raw concatenated dataset. Each of our four summarizers produced a summary for all examples. We calculated ROUGE scores between the reference and generated summaries and selected 4 ROUGE F1-scores (ROUGE-1, ROUGE-2, ROUGE-L, ROUGE-LSum) that show how good the generated summaries are. ROUGE tells us to what extent two pieces of texts overlap by counting n-grams (single words, word phrases of length 2 or more, or in the case of L, longest matching subsequence of words). We split data into train, validation, and test sets in ratios of 90:5:5.

[3] https://viri.cjvt.si/gigafida/
[4] https://www.clarin.si/repository/xmlui/handle/11356/1109

The sizes of both datasets are presented in Table 2. In Table 3, we present the average ROUGE values of our summarizers on long and short texts. Summarizers that are specialized for short texts achieve better results on short texts and vice versa.

Model	**Training size**
Doc2Vec	677,644
Metamodel	93,419

Table 2: Number of training samples for each model.

	t5-article	sumbasic	graph-based	hybrid-long
Short	**14,01**	13,11	13,15	12,55
Long	10,51	13,12	**17,71**	17,59

Table 3: Summarizers ROUGE scores for long and short texts. The best scores for short and long texts are in bold.

3 Summarization models

For this research, we used 4 summarization models. Sumbasic [4] uses a simple word frequency approach to select the most informative sentences. Graph-based summarization model [6] was inspired by the TextRank algorithm [2] and uses centrality scores of sentences to rank them. Both models belong to extractive methods and can be used on documents of any size. Our T5 abstractive summarization model (t5-article) uses a pre-trained T5 model [5] and continues training on a machine-translated CNN/Daily Mail dataset. Our hybrid-long summarization model is a combination of the graph-based and t5-article models. It first constructs a short text by concatenating the most informative sentences (extractive step). In the next, abstractive step, these sentences are summarized.

4 Method

To represent the meaning of a document, we used a Doc2Vec algorithm [1], an extension of Word2Vec [3]. In comparison to Word2Vec which uses only words inside a chosen window size, Doc2Vec adds an additional document token. Compare the figures 1 and 2. Like in Word2Vec, two versions exist: distributed memory (PV-DM) and distributed bag of words (PV-DBOW). PV-DBOW uses only

a document vector to predict the words inside a window. In contrast, PV-DM uses document vectors and word vectors to predict a single word. At the end of the training, this document vector is considered a representation of a document and can be used for many purposes, such as clustering, classification, etc. The main difference between Doc2Vec and Word2Vec lies in the inference step. There is no pre-calculated embedding for a document that was not part of the training phase. What we do is randomly initialize a document vector and continue training the model for a selected number of epochs while keeping other weights fixed. In other other words, we only make updates for a newly initialized document token. Usually, the same number of epochs as in the training phase is used.

In our work, we used a distributed memory version of the Doc2Vec algorithm with 100k *max vocabulary size*, *window size* 5, and *vector size* 256. *Max vocabulary size* determines the max number of words that will be stored and less frequent than that will be ignored. This helps to create a smaller model and ignore words without significant meaning, such as typos, extremely rare words, etc. *Window size* restricts the number of words we use for a single training example. With *window size* 5, we select 5 words before and 5 after a target word. In this sense, we get a lot of training samples from a single text.

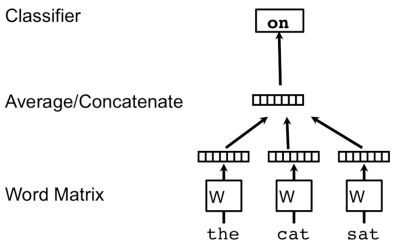

Fig. 1: Word2Vec. The algorithm predicts the word based on the current window of words.

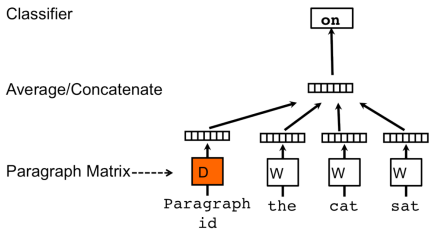

Fig. 2: Doc2Vec. The algorithm predicts the word based on the current window of words and a document token.

As our metamodel, we used a fully-connected two-layer neural network. The input consists of a 256-dimensional document embedding and the output of 16 ROUGE scores. Each of the two hidden layers has 768 *units* and a ReLU *activation function*. We decided to use mean squared error as our *loss function* and Adam as an *optimizer* with an initial *learning rate* of 0.001. During the training phase, we monitored validation loss and set *patience* to 2 which means that the network will automatically stop learning when validation loss has not improved for the last two epochs. To select the best model, we averaged predicted values (4 ROUGE scores introduced in Sec. 2) of each model and selected the model with the highest estimated final score.

The complete pipeline is presented in Fig. 3.

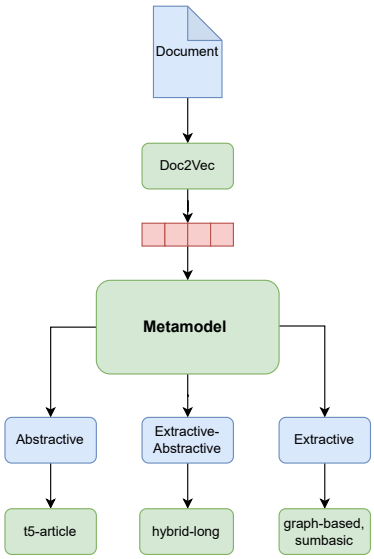

Fig. 3: The complete pipeline. Metamodel receives a vector representation of an input document from Doc2Vec and selects the most appropriate summarization model for it. Models are grouped by the type of output they produce: abstractive, extractive, and hybrid.

5 Results

We evaluated Doc2Vec using manual and automatic techniques. The manual analysis showed that using cosine similarity, the model successfully grouped documents by the same topic and that it works as expected. The automatic evaluation was part of the whole pipeline, where the model hyperparameters were tuned to optimize the loss of the metamodel.

Our final results are presented in Table 4. Mean-baseline is a model that simply averages all predictions for each summarization model. Metamodel stopped learning after 7 epochs and performed almost 15 points above Mean-baseline on the test set. We observed that choosing different hyperparameters does not seem to significantly affect the results. We experimented with different hidden layer sizes, numbers of units, and activation functions of metamodel. We also tried different max vocabulary and window sizes of the Doc2Vec model. The best parameters were reported in Section 4.

We experimented with two variations of the metamodel. Metamodel-length adds another input neuron that explicitly encodes the input length. We found that this does not improve the model and hypothesize that academic texts are of different genres and the document embedding technique covers it well already. We also tried to balance data since the original dataset contains a 1:5 ratio of long to short texts and rises a potential issue of overfitting on short texts. We reduced the number of short texts in a training set to get a balanced dataset of 16,932 samples for our Metamodel-balanced model. This resulted in a worse performing model but still better than Mean-baseline.

Model	**Mean squared error**
Mean-baseline	84.493
Metamodel-baseline	**70.066**
Metamodel-length	70.146
Metamodel-balanced	79.044

Table 4: Results of our four models on the test set. Metamodel-baseline showed significant improvement over Mean-baseline. Encoding length feature explicitly or balancing the dataset did not improve the results.

6 Conclusion

In this seminar, we developed a metamodel that automatically selects the most appropriate summarization model based on a neural document representation. We showed that our model improves the mean baseline. The model will be used to help users get the summary without manually selecting one of the models.

In the future, we could improve the document representation using more recent neural architectures such as transformers. User feedback can be used as well: users can select which of the 4 summaries is the best and this can later be used to optimize how ROUGE scores are combined (instead of simply averaging them). Instead of using raw ROUGE scores, we could transform outputs into the ordinal variable, and observe how this affects the whole pipeline.

References

1. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International conference on machine learning. pp. 1188–1196. PMLR (2014)
2. Mihalcea, R., Tarau, P.: Textrank: Bringing order into text. In: Proceedings of the 2004 conference on empirical methods in natural language processing. pp. 404–411 (2004)
3. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
4. Nenkova, A., Vanderwende, L.: The impact of frequency on summarization. Tech. rep., Microsoft Research (2005)
5. Ulčar, M., Robnik-Šikonja, M.: Sequence to sequence pretraining for a less-resourced slovenian language. arXiv preprint arXiv:2207.13988 (2022)
6. Žagar, A., Robnik-Šikonja, M.: Unsupervised approach to multilingual user comments summarization. In: Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation. pp. 89–98. Association for Computational Linguistics, Online (Apr 2021), https://aclanthology.org/2021.hackashop-1.13