Ways to summarize texts —

Szöveg összefoglalása különböző módszerekkel

Patrick Nanys
BME-VIK
Budapest, Hungary
patrick.nanys2000@gmail.com

Máté Jakab
BME-VIK
Budapest, Hungary
jakab.mate200@gmail.com

***Abstract*— In the field of informatics, deep learning technology has become one of the most popular topics of our time. We can find extremely interesting and useful results in many topics. The range of problems is wide, and new tasks are emerging day by day. In this paper, we have examined an older problem, the summarization of texts. We collected a number of data processing technologies, applied stemming, lemmatizing, and also checked the results by taking out the stopwords. The goal of our solution was to create a model that is capable of abstract summarization. We used a supervised learning database to train our model, the model was LSTM based. We tested popular and well-functioning technologies on the topic (GloVe, BERT) and also examined their effectiveness. The best results were obtained by fine-tuning the BERT pretrained model. Unfortunately, we did not achieve a widely useful model with high-quality text summaries in our work, but it was successful to learn about the technologies and master their practical application.**

***Összefoglalás*—Az informatika területén a mélytanulásos technológia korunk egyik legnépszerűbb témakörévé vált. Számos témakörben rendkívül érdekes és hasznos eredményeket találhatunk. A problémák tárháza széleskörű, és napról napra új megoldandó feladatok alakulnak ki. Jelen dokumentumban egy régebbi problémát vizsgáltunk meg, a szövegek összegzését. Összegyűjtöttünk számos adatfeldolgozási technológiát, alkalmaztuk a stemminget, lemmatizingot, és ellenőriztük az eredményeket a stopword-ök kiszedésével is. A megoldásunkban egy absztrakt összefoglalásra képes modell készítése volt a cél. A modellünk tanításához egy kimeneti értékekkel ellátott adatbázist használtunk (supervised learning), a modell LSTM alapú volt. Kipróbáltuk a témában népszerű és jól működő technológiákat (GloVe, BERT) és megvizsgáltuk a hatékonyságukat is. A legjobb eredményeket a BERT előtanított modell fine-tuningjával értünk el. A munkánkban sajnos nem értünk el széles körben használható, magas minőségű szövegösszefoglalásra képes modellt, viszont a technológiák megismerése és gyakorlati alkalmazásának elsajátítása sikeres volt.**

Keywords—neural network, NLP, text summarization, text preprocessing, LSTM, BERT, GloVe

I. Introduction

Nowadays, due to the possibilities provided by networks, communication has increased enormously, so the amount of data has increased enormously, and managing and analyzing have become difficult. This digitalization has had an impact on our daily lives for years, but due to COVID-19, which has been in the public domain for a year and has been causing problems (e.g. complete lockdowns), even more data has flowed on the Internet. Companies aim to use this data, but according to a 2016 article, 60-73% of these data within companies are wasted analytically and are not used for analysis.
Of the many types of data, they all need to be handled in different ways, but the range of possibilities is almost endless. In this paper, we have been working on word processing and text summarization. Our goal was to create a model that returns a few words of summary as an output if we enter a short text. We used different word processing technologies in order to get the most accurate model possible.

The number of purchases on the web has greatly increased in the last year, which has the advantage of not having to leave the apartment when shopping, but the disadvantage is that we can't take a closer look at the product. For this reason, we have to rely on information on the Internet, a large percentage of which is the opinions of previous customers. Many customers are happy to share their opinions about the product, rate it on a five-point scale, or even make a review. Based on the content of these text notes, the benefits and even the disadvantages of the products can be summarized. This compiled database can be useful for an online store that is unaware of all the products sold, their advantages and disadvantages. You can use it to promote the product by showcasing the benefits, encouraging more people to buy.

In our solution, we used the reviews of Amazon's web store and their summaries to train a model that is able to summarize texts in an abstract way. The abstract in this case means that it uses completely new words for the summary, in contrast to the extractive summary, where it finds some sentences and expressions that are considered more important in the input text. The very first step of our solution was, of course, data preprocessing, because the words had to be brought into a form that could be understood by the machine in some way. We tested the effectiveness of omitting words with little information (extracting stopwords), examined the difference between stemming and lemmatizing and tried them out in our

preprocessing method. Our first model was an LSTM-based neural network. We wanted to improve its precision, so we checked other ways of creating the model. We examined the performance of the model by using a common pre-taught GloVe embedding instead of separate encoder embedding and decoder embedding. We also examined the results with a model that is pretrained on BERT.

II. Related work

A summary of the texts can also be useful for additional tasks, such as categorizing emails to a company's email address. In this case, the short summary will be one or two words, i.e., the task changes to a categorization task, not summarization. An example of this can be found in the following document: [1]

Preprocessing of texts is the first task in an NLP system. We implemented a lemmatizer in our solution (more about its usefulness: [2]) and we also tried stemming and even took out the stopwords. There are some other methods (eg. multiword grouping), which can be read in detail in the following paper: [3]

An important subtask of preprocessing is tokenization, there are several types of it: word tokenization, subword tokenization, and character tokenization. In our solution we used the word tokenization. A simple example of subword tokenization, and its usefulness is presented in the following publication: [4].

There are two main directions for summarizing texts, the abstractive and the extractive. In our solution, we implemented an abstract summarization, ie the model can give words that are not necessarily found in the original text. This can be used, for example, to give names of shorter articles that you want to publish. In the case of the extractive summarization, the model highlights the most important parts of the text, an example of this technology can be found in the following publication: [5]. The abstractive summarization is considered as a more difficult task, an example of using this technology is in this paper: [6].

In our solution, the abstract text summarization was provided by learning using a database where the output values (labels) were available, i.e., the learning was supervised. There are unsupervised solutions as well [7], where training takes place without the help of predefined labels.

Our model provided few words as output, but there are several models that can provide multi-word or even multi-sentence summaries as output. An example of this is in the following publication [8], where they will also provide a new database for those who want to do further research in the field of multi-sentence summarization.

Summarization of neural abstract text has gained high popularity with sequence-to-sequence (seq2seq) models. Many interesting techniques have been proposed to improve the development of seq2seq models that can handle various challenges such as fluency and human readability. For more information on the seq2seq models, we recommend the following summary: [9].

III. System design

In our solution, we examined three models in terms of performance. The first version was an LSTM model. We created the model with the help of the following articles:[10], [11]. To create the second version, we used GloVe Embeddings [12]. Lastly, for the third version, we used BERT [13]. We present both the preprocessing and the training process.

For the architecture we chose to go with a sequence to sequence model that later on we enhanced with the attention mechanism to improve the models ability to see the input text as a whole and understand the connections between the individual words better.

Our model is made up of an encoder and a decoder.

For the encoder we used an embedding to map the input words into a dense representation of themselves and then used three layers of LSTMs to extract the information out of the input. We chose to go with LSTM layers because of their well-known ability to effectively pass through information even when using multiple layers.

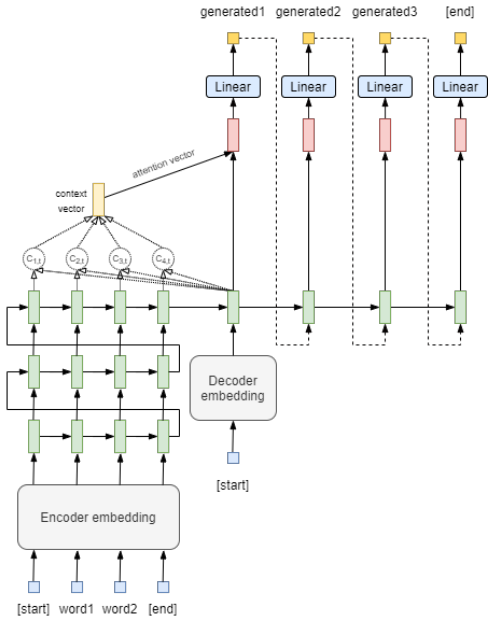
In the decoder we run the decoder inputs through an embedding and a single LSTM layer and then we use the outputs of the encoder as the query and the outputs of the decoder LSTM layer as the key argument of the Attention layer to calculate the context vector for a given word. Then we concatenate the output of the decoder LSTM and the Attention layer and pass it through a linear layer at the end to classify which word the model predicts next. Concatenation is used to make sure that the linear model has all the meaningful information from which it could learn from to make sure it is making an informed prediction.

While the encoder takes a whole text as its input all the time, the decoder takes a whole text as an input during the training phase and takes one word at a time during generation.

A. Generation

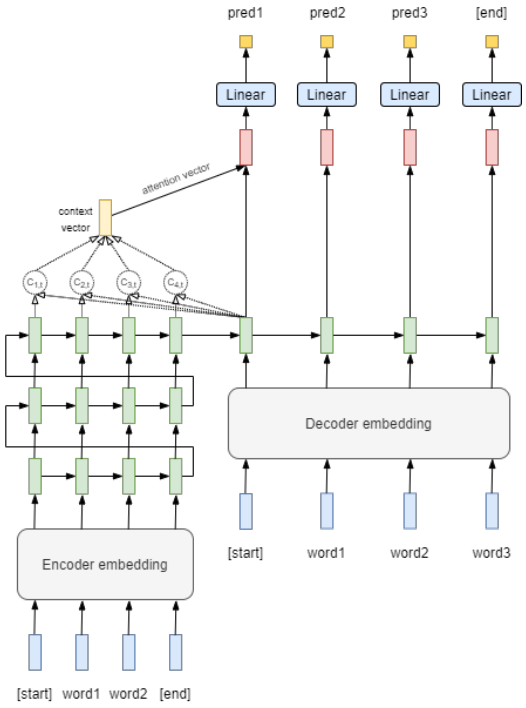
Basically, similar preprocessing was used for all three models, with differences in naming. In our simplest model (Simple LSTM), the sequences contained the [start] and [end] in the beginning and in the end of a sequence, as you can see in 1. Figure.

This was changed to [CLS] and [SEP] in the BERT-based model, and [UNK] tokens were at both at the beginning and end of a sequence in the GloVe-based model.

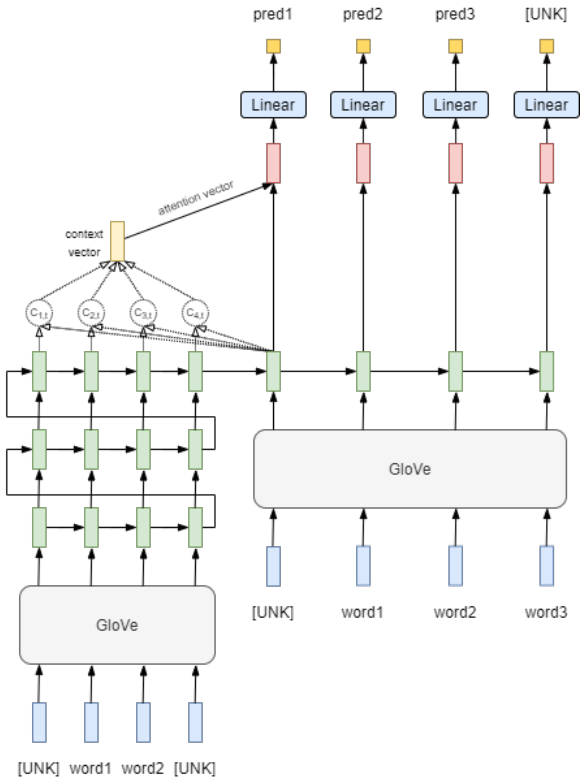


1. Figure –LSTM model

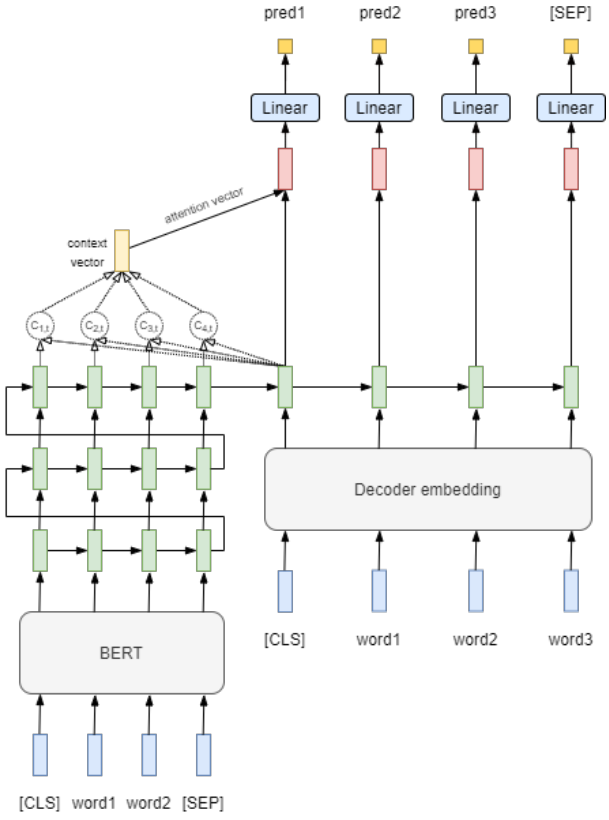
B. Training



2. Figure - Simple LSTM model



3. Figure - GloVe with LSTM model



4. Figure - BERT with LSTM model

IV. Implementation

We implemented our models in python, we used Google Colaboratory because of free access of resources. In this chapter we present the process of data acquisition and preprocessing and training. We describe the achieved results and test the obtained model.

A. Acquisition and preparation of data

We used the Amazon webshop reviews database available at the link below [14], the processing of which has been divided into the following parts:

1. Database loading
2. Taking out stopwords
3. Shorten words
    a. Lemmatizing
    b. Stemming
4. Tokenization
5. Padding
6. Split dataset

The database was loaded from an .xls file. Firstly, we have removed all the words in the input database texts that do not provide additional information. Examples of such words are *the, a, an, in,* and so on. If a word in the database was not a stopword, then in one version (*3.a*) we tried lemmatizing, which reduced the amount of conjugated words by assigning the dictionary version to the words, e.g. *is, are, am* are also set to *be*. In the other version (*3.b*), we tried stemming, which is the process of reducing words to their root form. These will not necessarily be meaningful words, e.g. the words *trouble, troubling and troubled* are also set to *troubl*.

After taking out stopwords and shortening remaining words in the sequences, we organized each word into separate units. This process is called tokenization. Tokenization had to be performed on both input and output texts. After tokenization, it was necessary to make each input and output sequences to the same length. To reach this goal, we set a maximum sequence length for both input and output texts. The maximum sequence length is equal to the length of the longest sequence of the first 75% of the words in ascending order of size. This way, the variance of the database will not be too large. The database was then divided into a training, validation and testing database in 0.8, 0.1, 0.1 proportions.

In contrast to the simple LSTM, and LSTM with glove models, we used BERT's own tokenizer for tokenizing the BERT-based model, this was necessary so that the pretrained model could handle the data.

B. Training

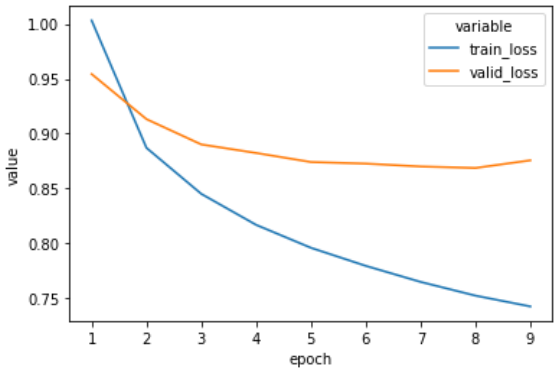
Due to the large number of parameters in the LSTM model, the training time is high, which is why we also needed to reduce the size of the training database so that the Google Collaboratory session does not end automatically before saving the model. During the evaluation, we examined the results for 10 epochs (with 32 batch size) and used early stopping (with 3 patience). We used rmsprop optimizer, and sparse categorical crossentropy as the loss function since it converts the integer sequence to a one-hot vector on the fly, which is overcomes any memory issues.

Params of LSTM model were the same

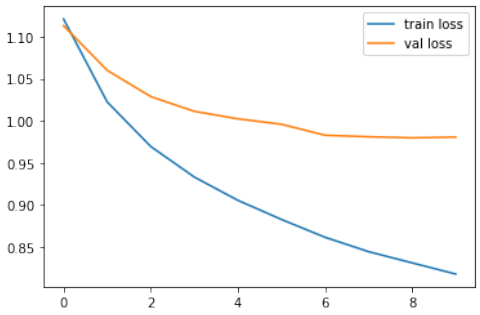
- embedding dim = 100
- hidden dim = 250
- enc lstm num = 3
- enc lstm dropout = 0.4
- enc lstm recurrent dropout = 0.4

C. Evaluation

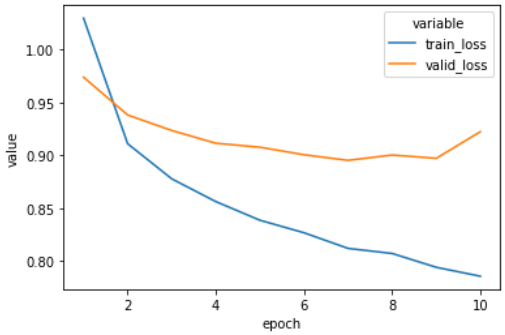
We can check train and validation loss in these figures: 5. Figure, 6. Figure, 7. Figure



5. Figure – LSTM model



6. Figure – LSTM with BERT



7. Figure – LSTM with GloVe

We did not calculate such metrics at the end of training to see exactly how good our model is, because we did abstract text

summarizations that can use completely new words, so it is hard to decide if a prediction is wrong or right.

We tried to calculate a rouge score on them (which essentially indicates how many words are the same in the original text than in our output), but since it is an abstractive summarization, it did not give meaningful results. More about rouge value: [15]

D. Testing

Due to the problems stated above we could not calculate any metrics that would represent the performance of our model well enough that's the reason we only provide the comparison of the outputs of the different models we experimented with. The following section presents the results obtained for the different models.

	Original title	**only LSTM**	**BERT**	**GloVe**
1	wonderful tasty taffy	great taste	great taste and good for you	great taste
2	my cats love this diet food better than their regular food	great food	good product	great food, but not as good as the cats
3	my cats are not fans of the new food	good for cats	good for cats	great food, but not as good as the cats
4	cough medicine	good but not as good as the bit own	great taste	great product
5	not as advertised	not as good as the product	not what i expected	not as good as the money

The table shows some examples, suggesting that the model is not able to find the exact meaning of the texts at a high rate. The summary is short in each case, just a few words, and it can be said that models often figure out the topic of the text (e.g., related to cats, related to food). Unfortunately, the model almost always uses positive words (most often the adjectives great and good). We are not necessarily happy with such optimism, as it often predicts a positive summary for negative reviews too. One of our favorite results reviewed was the prediction of the GloVe-based model in the last row. The "not as advertised" label has almost same meaning than "not what i expected".

V. SUMMARY, FUTURE WORK

In our work, we were able to become acquainted with the types of word preprocessing mechanisms with a neural network. From the repository of preprocessing operations, we tested our model with several methods (lemmatizing, stemming, removing stopwords). We built an LSTM-based model that unfortunately predicted the summary of opinions with low accuracy. We examined the performance of the model by using GloVe embedding instead of separate encoder embedding and decoder embedding. By fine-tuning the BERT pretrained model, we were able to test the results on another model.

Unfortunately, we did not get accurate results, it may be worth working to improve it. Modifying the parameters of the LSTM model did not improve our results, so the error is probably not to be found here. Unfortunately, due to the high number of parameters and the large database, training was an extremely long process and our resources were limited. To reduce training time, we reduced the size of the database so we taught the model with much less data, which could have caused a decrease in performance. We will try training with a full database on a stronger computer.

[1] D. K. Gupta és S. Goyal, „Email Classification into Relevant Category Using Neural Networks", *arXiv:1802.03971 [cs]*, febr. 2018, Elérés: dec. 11, 2020. [Online]. Elérhető: http://arxiv.org/abs/1802.03971.

[2] A. Kutuzov és E. Kuzmenko, „To lemmatize or not to lemmatize: how word normalisation affects ELMo performance in word sense disambiguation", *arXiv:1909.03135 [cs]*, szept. 2019, Elérés: dec. 11, 2020. [Online]. Elérhető: http://arxiv.org/abs/1909.03135.

[3] J. Camacho-Collados és M. T. Pilehvar, „On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis", *arXiv:1707.01780 [cs]*, aug. 2018, Elérés: dec. 11, 2020. [Online]. Elérhető: http://arxiv.org/abs/1707.01780.

[4] T. Kudo és J. Richardson, „SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing", *arXiv:1808.06226 [cs]*, aug. 2018, Elérés: dec. 11, 2020. [Online]. Elérhető: http://arxiv.org/abs/1808.06226.

[5] V. T. Chou, L. Kent, J. A. Góngora, S. Ballerini, és C. D. Hoover, „Towards automatic extractive text summarization of A-133 Single Audit reports with machine learning", *arXiv:1911.06197 [cs, stat]*, nov. 2019, Elérés: dec. 11, 2020. [Online]. Elérhető: http://arxiv.org/abs/1911.06197.

[6] M. E. Johnson, „Automatic Summarization of Natural Language", o. 6.

[7] Y. Jernite, „Unsupervised Text Summarization via Mixed Model Back-Translation", *arXiv:1908.08566 [cs]*, aug. 2019, Elérés: dec. 11, 2020. [Online]. Elérhető: http://arxiv.org/abs/1908.08566.

[8] R. Nallapati, B. Zhou, C. N. dos santos, C. Gulcehre, és B. Xiang, „Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond", *arXiv:1602.06023 [cs]*, aug. 2016, Elérés: dec. 11, 2020. [Online]. Elérhető: http://arxiv.org/abs/1602.06023.

[9] T. Shi, Y. Keneshloo, N. Ramakrishnan, és C. K. Reddy, „Neural Abstractive Text Summarization with Sequence-to-Sequence Models", *arXiv:1812.02303 [cs, stat]*, szept. 2020, Elérés: dec. 11, 2020. [Online]. Elérhető: http://arxiv.org/abs/1812.02303.

[10] R. C. Staudemeyer és E. R. Morris, „Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks", *arXiv:1909.09586 [cs]*, szept. 2019, Elérés: dec. 11, 2020. [Online]. Elérhető: http://arxiv.org/abs/1909.09586.

[11] „Text Summarization | Text Summarization Using Deep Learning", *Analytics Vidhya*, jún. 10, 2019. https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/ (elérés dec. 11, 2020).

[12] J. Pennington, R. Socher, és C. Manning, „Glove: Global Vectors for Word Representation", o. 12.

[13] J. Devlin, M.-W. Chang, K. Lee, és K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *arXiv:1810.04805 [cs]*, máj. 2019, Elérés: dec. 11, 2020. [Online]. Elérhető: http://arxiv.org/abs/1810.04805.

[14] „Amazon Reviews for Sentiment Analysis". https://kaggle.com/bittlingmayer/amazonreviews (elérés dec. 11, 2020).

[15] K. Ganesan, „ROUGE 2.0: Updated and Improved Measures for Evaluation of Summarization Tasks", *arXiv:1803.01937 [cs]*, márc. 2018, Elérés: dec. 11, 2020. [Online]. Elérhető: http://arxiv.org/abs/1803.01937.