NLP challenge - Spain AI

IE University Default Project

Sergi Abashidze
Bachelor of Data and Business Analytics
IE University
sabashidze.ieu2018@student.ie.edu

Paula Garcia Serrano
Bachelor of Data and Business Analytics
IE University
pgarcia.ieu2018@student.ie.edu

Reem Hageali
Bachelor of Data and Business Analytics
IE University
rhageali.ieu2018@student.ie.edu

Sidhant Singhal
Bachelor of Data and Business Analytics
IE University
ssinghal.ieu2018@student.ie.edu

Abstract

Text summarization is the process of producing concise summaries while preserving significant information. In this paper, we investigate the issue of summarizing product titles for Zara given the product description. With the increase in the e-commerce platforms, it is necessary to compress products' titles for consumer's convenience. Throughout this paper, we evaluated 5 different models, and the best one turned out to be the T5ForConditionalGeneration, a state of the art pre-trained generative model. It is an encoder-decoder model and converts all natural language processing problems into text-to-text format. However, the data at hand had a few factors that impacted the model performance, resulting in poor predictions. The product titles included some words that were not present in the description, and there were also some spelling errors in both the product name and description. We were able to tweak the model to gain predictions that were satisfactory after many iterations.

The reader can find how we executed this project in the following link: `https://github.com/paulagarciaserrano/NLP_Zara_Challenge`

1 Introduction

This paper describes the problem that the authors tried to solve when facing the Spain AI Zara Challenge. The task in this hackathon was to generate product names given the product description. This task is interesting because product name is one of the most important factors the consumer considers before making a purchase, so we wanted to make sure that using natural language processing and data-oriented solutions we were able to generate names which will adequately encompass the product's description. However, after inspecting the data at hand, we became aware of a few factors that were going to impact our performance. First of all, in the training set of the data, we spotted some words present in the product's name that did not appear in the product's description. This accounted for 16% of the tokens, and 13% of the training observations were affected. Not only this, but the team also spotted some typos in the words conforming the product's name and description ("with" appeared written as "whit"). Therefore, taking into account that the competition guidelines required an exact match (which included spaces, hyphens, and parentheses) to consider a prediction as a right one, we soon acknowledged that our performance in this hackathon would hardly reach the expected standards.

2 Related Work

Interest in text summarization dates back to the 1950s. A paper was published in 1958, based on a statistical approach, were the suggested method was to weigh each sentence in a document as a function of high-frequency words and ignoring the common words that have a very high frequency. Research increased significantly throughout the years, what resulted in developing new techniques.

There are two forms of solving a text summarization problem. The first of these, the extractive, consists on selecting significant sentences from the document and forming them into a summarized sentence. This is formulated by taking certain text segments from the document which are measured by statistical analysis of certain features such as word frequency or keywords to spot significant sentences. The second of these, the abstractive, entails paraphrasing and shortening parts of the source document, to create new phrases and sentences that relay the most useful information from the original text.

Most work on text summarization has been focused on extractive-based methods. Rasim Alguliev proposed to solve the problem by having an unsupervised document summarizing approach which will develop the summary by extracting and clustering phrases from the original document. A discrete differential evolution algorithm was developed to reach optimal criterion functions. The results are shown on a benchmark dataset from DUC2001 and DUC2002, which claim that their approach could improve the performance compared to the top summarization approaches [1]. Joan Xiao and Robert Munro tried a different approach and compared two extractive approaches using a bi-directional long short-term memory encoder-decoder network with an attention mechanism. This method is used to address sequence to sequence (seq2seq) problems, it is challenging due to the varying number of inputs and outputs. They tested this by first presenting a multi-class named entity recognition approach, and then testing a binary class named entity recognition to measure the best solution to the problem using ROUGE-1 and ROUGE-2 as a metric [2]. In natural language processing, ROUGE is a set of metrics and a software package used for testing automatic summarization and machine translation. ROUGE-1 refers to the overlap of each word (unigram) between the system and reference summary, and ROUGE-2 refers to the bigrams between the system and reference summary [3].

On the other hand, regarding abstractive summarization we find sequence to sequence models, that converts sequences from one domain to sequences in another domain. The model consists of 2 parts: encoder, and decoder. Both are an RNN layers, the encoder processes the input sequences and returns their internal state, it serves as the context of the decoder which is trained to predict the next word of the target sequence [4]. Nallapti et al uses abstractive text summarization using the attentional encoder-decoder recurrent neural network and with large vocabulary trick on two different corpora which was developed as a machine translation model. The encoder has a bidirectional GRU-RNN while the decoder is unidirectional, but has the same hidden-state size as the encoder, with an attention mechanism over the source hidden-state, and a soft-max layer over the target vocabulary. GRU is similar to LSTM with a forget gate but has fewer parameters since it does not have an output gate. With this model, the most frequent words will be added to the vocabulary until it reaches its limited size. This technique will reduce the size of the soft-max layer since it can hold up the system, it is a bottleneck while computing [5].

Pre-trained models have advanced in many natural language processing tasks from sentiment analysis to named entity recognition. Recently a pre-trained model called BERT applies bidirectional training of Transformers — an attention mechanism that understands the relations between words. Yang Liu and Mirella Lapata showcase BERT in a general framework for both extractive and abstractive models and introduce a novel document-level encoder. The author evaluates the approach on three documents news summarization datasets, with different writing and summary styles. The results across the three datasets showed that the model is a state-of-the-art under automatic summarization and human-based evaluation protocols [6]. Another pre-trained model is BART, which uses a standard seq2seq architecture with a bidirectional encoder such as BERT, and a left-to-right autoregressive decoder such as GPT. Lewis et al presents BART, a denoising autoencoder for pretraining seq2seq models. For the base model, 6 layers in the encoder and the decoder, and for the bigger model 12 layers are used in each. Evaluated the model on different tasks: sequence classification, token classification, sequence generation, and machine translation. By adding a new set of encoder parameters, it was possible to use the BART model as a whole as a single pre-trained decoder for machine translation [7].

3 Experiments

3.1 Data

The competition documentation consists of two datasets. A training dataset, consisting of two columns: product name, and product description; and a testing dataset containing only the product description column. Both of these datasets are in English. Moreover, the training dataset consists of 33613 observations, whereas the testing dataset contains 1441 observations. As explained previously, the task with this data was to use the product description as input, to generate the product name as output. After performing the exploratory data analysis for the training dataset we discovered two main issues. Firstly, there were 16% of the product name tokens that were not present in the product descriptions, and 13% of the training observations were affected by this. Secondly, there are some tokens that do not make a lot of sense, as for example: *BLT 05*, *TRSR CMB 05*, or even *BLT BG SRPLS @*. These create big difficulties for us since the predicted product name must be an exact match in order to get points, and to win the competition.

3.2 Evaluation method

In order to evaluate our predictions and modelling performance, we are going to rely on Recall-Oriented Understudy for Gisting Evaluation (ROGUE) metric. ROGUE is a set of metrics and a software package used for evaluating automatic summarization and machine translation software in natural language processing. The metric compares an automatically produced summary or translation against a reference or a set of references (human-produced) summary or translation.

The ROGUE metric consists of:

- **ROGUE - N**: measuring the number of matching 'n-grams' between our model-generated text and a 'reference'
- **ROGUE - L**: measuring the longest common subsequence (LCS) between our model output and reference
- **ROGUE - S**: allowing us to search for consecutive words from the reference text, that appear in the model output but are separated by one-or-more other words.

For each ROGUE, the following is computed:

- Recall: counting the number of overlapping n-grams found in both the model output and reference, divided by the total number of n-grams in the reference
- Precision: which is calculated the same way as recall but rather than dividing by the reference n-gram count, it is divided by the model n-gram count
- F-1 Score: which is the harmonic average of the precision and recall

3.3 Experimental details

Throughout the process, we tried five different methods and evaluated which displayed the best performance. We first focused our attention on encoder-decoder models. The first model that was attempted was an encoder-decoder using LSTMs. We tried creating our own neural network, mainly using the TensorFlow library, and with the following layers, in this order: one encoder with embedding layer, three LSTMs, one decoder with embedding layer, one LSTM, one attention layer, and one dense layer. We tried the 'rmsprop', 'Adam' and 'SGD' optimizers, and used the 'sparse_categorical_crossentropy' as the loss function. As we saw that this model was not bringing good results, we moved on, and tried a second encoder-decoder, but this time with more layers. For this, we used the torch and TensorFlow libraries, and built our model following this layer architecture: one encoder with embedding layer, one GRU, one decoder with embedding layer, one dropout, one linear, one softmax, one linear, one ReLu, one GRU, one linear, and one softmax. We implement it with the 'SGD' optimizer and 'NLLLoss' as the loss function. Lastly, as this did not work either, we moved to use pre-trained models. Therefore, we continued with the encoder-decoder strategy using Bert. For this, we relied on the torch and transformer libraries and chose 'Adam' as the optimizer.

Unfortunately, these approaches using these encoder-decoders did not work, so we moved on to try different models, and we tried the GPT-2 model from the gpt_2_simple library. To do this, we chose the model size to be 124M, and the optimizer to be 'AdamW'. But it did not work either.

Lastly, we tried our last attempt, and chose another pre-trained model, T5 for Conditional Generation, using the torch and transformers libraries. For this model, we used the 'SGD' optimizer, and finally, were able to get some decent results.

3.4 Results

As for the results, as explained in the section above, we need to say that nearly none of the models were considered to be even performing. Regarding the first model described, the encoder-decoder using LSTMs, this showed no learning at all, as the loss of the model would not decrease throughout the iterations. Something similar happened with our second experiment, an encoder-decoder with more layers, as it did not learn either.

Furthermore, regarding the third experiment, the Bert encoder-decoder, this seemed to learn, but when predicting, it was only able to predict the first token of the sentence, and all of the rest were constant, and not correct. Moving onto the next pre-trained that we tried, we are not able to report results, as we did not have the computational power to evaluate this model, the expected evaluation time consists of 40 hours.

Lastly, regarding our fifth experiment, the pre-trained T5, it was the the only model that we computed the Rouge score for, as it was the only one we could correctly train. The results for this were:

- ROGUE-1: {f: 0.21, p: 0.23, r: 0.24}
- ROGUE-2: {f: 0.05, p: 0.05, r: 0.06}
- ROGUE-L: {f: 0.21 , p: 0.23, r: 0.23}

4 Analysis

Previous to analyzing the results, the team thinks that it is important for the reader to notice that for the cases where tokens occurred once or twice in the whole product description collection (e.g. *Disney*), we decided to not include them in the vocabulary, as this was biasing the results.

Generally speaking, the model was able to make predictions; in many instances, it predicted the exact word which had to be predicted just like in the test dataset, yet as for the overall structure of the predicted sentence, the model failed to predict the exact match. Based on the scoring, from the ROGUE metric, we can see that the overlaps of unigrams (ROGUE-1) and bigrams (ROGUE-2) of the predictions were poor compared to the actual results.

During the modelling, we have come across the problem of overtraining. Moreover, comparing the predicted product name to the actual one, we saw that our model had difficulties predicting long text.

5 Conclusion

Working with pre-trained models we have come across with two main issues. The first one was that we had to deal with the scarce computational resources that we have. Since the pre-trained models had millions of parameters we had to use Google Colaboratory in order to fit the model which overall took hours for the training process. Sometimes, this was not enough, and so we decided to rely on the cloud services from Azure. Working on this project we have understood how important it is to have enough computational capacity in order to get adequate results and complete this kind of projects. The second issue we came across with was the over-training done by these models which was challenging to deal with.

In addition, even working with state-of-the-art models, our predictions were poor due to the small size and poor structure of the dataset that was provided to us. We saw instances where we had to predict a word that was not even mentioned in the product description column. These types of issues made the project even more difficult. That said, we understood how important it is to have good, structured and valuable dataset since the model will never predict words which it has never learned.

Given this situation that we encountered, we had major difficulties making predictions even using state-of-the-art models like GPT-2, BERT and T5. However, after many iterations and tweakings done to the models, we were able to get predictions that satisfied us.

Despite poor results (from ROGUE metric), as a team, we are well pleased with our work since this was the first time for us using state-of-the-art methods in Natural Language Processing, and in machine learning overall. The challenges we faced during the completion of this project made us think outside of the box and implement things that we have never done before.

References

[1] R. ALGULIEV. Evolutionary algorithm for extractive text summarization. In *Intelligent Information Management. 1st ed.*, pages 128–138, 2009. Available at: `http://www.scirp.org/journal/iim`.

[2] J. Xiao and R. Munro. Text summarization of product titles. Paris, France, 2019. SIGIR. Available at: `http://ceur-ws.org/Vol-2410/paper36.pdf`.

[3] K. Ganesan. An intro to rouge, and how to use it to evaluate summaries. *freeCodeCamp.org*, 2017. Available at: `https://www.freecodecamp.org/news/what-is-rouge-and-how-it-works-for-evaluation-of-summaries-e059fb8ac840/`.

[4] F. Chollet. A ten-minute introduction to sequence-to-sequence learning in keras. *Blog.keras.io*, 2017. Available at: `https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html`.

[5] Zhou B. dos Santos C. Gulçehre Ç. Nallapti, R. and B. Xiang. *Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond.* 2016. Available at: `https://arxiv.org/pdf/1602.06023.pdf`.

[6] Y. Liu and M. Lapata. *Text Summarization with Pretrained Encoders.* 2019. Available at: `https://arxiv.org/pdf/1908.08345.pdf`.

[7] Liu Y. Goyal N. Ghazvininejad M. Mohamed A. Levy O. Stoyanov V. Lewis, M. and L. Zettlemoyer. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.* 2019. Available at: `https://arxiv.org/pdf/1910.13461.pdf`.