AutoWiki - Final Project

Edmar Rodrigues Filho, Gabriel Oliveira dos Santos

December 2021

Abstract

Nowadays, we use search engines every day to navigate on the Internet. However, these mechanisms currently return us a list with the pages that we are more likely to be interested in. It makes the navigation inefficient because we need to read several pages to find out the complete information we are searching for. Thus, an intelligent engine would be capable of creating a new page containing the complete information a user searches for. In order to start the development of an intelligent engine and considering the scope of this course, we propose a model that predicts the sections of a Wikipedia page given the title and the abstract. This model can then incorporated into more complex systems which can generate a complete Wikipedia page.

1 Introduction

Search engines are essential tools for the successful navigation of the growing online informational world. Nowadays, we can search for any terms, and our search engine will return us a list with the pages most probable to be related to the query and we which are more likely to be interested in. However, the information is dispersed across different pages, and documents can be outdated. These factors contribute to the inefficiency of the current search engines in returning results that quickly respond to user queries.

In light of this problem, aiming to provide more intuitive and relevant results to users' queries, we explored generative models based on T5[4]. Our models predict the sections of a Wikipedia page given the title and the abstract. It is the first step in developing a system capable of generating a complete web page. This way, a search engine will be able to generate a new page in order to answer the user's question, summarizing the most relevant information that was originally dispersed throughout the Web.

2 Methodology

To tackle the problem of predicting sections of a Wikipedia page given the title and the abstract, we trained seq2seq models based on T5[4]. To train our models

we used the TREC Car[1] dataset, versions 2.0 and 2.1. However, this dataset does not contain the abstract text of the Wikipedia pages.

To do so, we had to retrieve this information from the Wikipedia API, we illustrate this process in Figure 2. For each page in TREC Car dataset, we search for its title via Wikipedia API and we try to retrieve the page content. If the page is found, then we store the abstract together with remaining page texts. But, there are some pages that are not found by Wikipedia API, because the title can be outdated. Also, there are a few pages that do not contain sections. In these two cases, we ignore the pages.

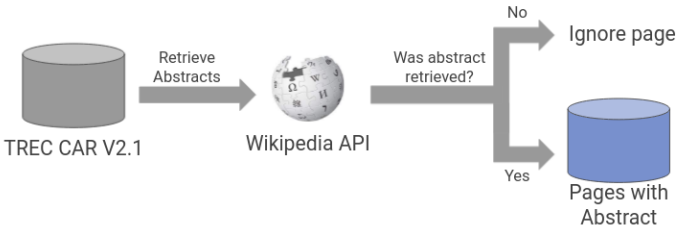

Figure 1: Pipeline for retrieving abstract from Wikipedia pages.

Moreover, we preprocessed the texts so that our models' input is the title concatenated with the abstract, separated by a [SEP] token. The outputs are composed of section titles also separated by a [SEP] token.

Finally, to measure the quality of the generated texts, we used the F1-Score considering each title as a single element of the sequence. When the number of predicted sections is different from the original number of sections in the page, we add a blank titles so that the lists of original and predicted sections have the same size. In addition, we used the BLEU[3] metric to measure more qualitatively the generated texts.

3 Data set

In order to train our models, we use the TREC Car V2.0 and V2.1[1] dataset. It is composed of Wikipedia pages, with title and section contents, but not the abstract. In particular, we used `train.v2.0` set to train our models, and `benchmarkY1-test-public.v2.0` and `benchmarkY2test-public.v2.1.1` sets to test them. Since the dataset does not contain the text from the abstract, we had to retrieve this information from the Wikipedia API, as described in Section 2.

After the retrieving process, we end up with 285,719 pages in the train set, and 60 and 31 from `benchmarkY1-test-public.v2.0` and `benchmarkY2test-pub lic.v2.1.1`, respectively. Thus, we used split the pages from `train.v2.0` into train and validation set, according to the proportion 80% and 20%, respectively. Still, we used both `benchmarkY1-test-public.v2.0` and `benchmarkY2test-pub`

`lic.v2.1.1` sets to test our models, because they consist of manually curated pages.

4 Experiments

In order to train our models to predict the section contents, the inputs were created by concatenating the title with the abstract, separated by a [SEP] token. The outputs comprehend concatenated section titles separated by [SEP] tokens. We carried out experiments with different sizes of the T5 model, varying the maximum sequence lengths for both input and output. In our experiments, we used the Adam optimizer, and we fixed the batch size in 64, except for the model with maximum sequence length equal to 256, where the batch size was 32. Also, aiming to explore different decoding methods, we tested the Greedy Search and Beam Search approaches. Still, to avoid the problem of the model repeating a short sequence of words as pointed out by Vijayakumar et al. [5], we also experimented the method Top-p (nucleus) sampling [2], which was proposed to improve the text diversity.

4.1 Quantitative Analysis

Table 1 shows the results of our models on validation and test sets.

GPUs	Model	Max. Length	Validation		Test	
			BLEU	F1-Score (%)	BLEU	F1-Score (%)
1 Tesla P100	t5-base	128	30.99	15.8	16.50	2.5
	t5-small	128	28.11	15.7	14.22	3.6
2 Titan XP	t5-small	128	16.60	8.7	13.43	2.0
	t5-small	256	12.02	8.1	7.56	1.9
	t5-small - Beam Search	128	7.92	5.7	2.16	2.1
	t5-small - Top-p	128	12.61	1.6	9.62	0.8

Table 1: Performance of models on validation and test set. By default, the decoding method is Greedy Search, except when explicitly mentioned on the Model column.

Overall, we can see that the t5-base model has the best performance on predicting sections, as indicated by the BLEU score. However, the performance is not significantly hampered when we use t5-small, so it can be a better option when memory is limited. In addition, it is worth noting that when we used two GPUs Titan XP instead of one Tesla P100, the results drop considerably on the validation set even though all hyperparameters remain the same. We hypothesise that it occurs because of the way the batches are made parallel, which can interfere with BLEU and F1-Score calculations. However, note that the performances of the t5-small model trained using one and two GPUs are similar on the test set.

Moreover, we carried out experiments varying the maximum sequence length, we considered maximum lengths of 128 and 256 tokens. We can see that the model using 128 tokens has better results than the one using 256 tokens. As

illustrated in Section 4.2.2, increasing the maximum sequence length allows the model to repeat more tokens and, thus, get a lower BLEU score. However, both models have similar F1 scores, which means that they predict all the sections correctly in the same proportion.

Regarding the different decoding algorithms, we can note that greedy search is the one that has the best results, whereas beam search is by far the worst one, considering the BLEU score. Nevertheless, when we consider the F1-score, the beam search method surpasses the Top-p sampling approach. It can be explained by the fact that Top-p sampling aims to improve the diversity of the generated texts while beam search tries to find the most likely sequence of tokens. Thus, Top-p sampling rarely generates the same section title as the original, resulting in bad F1 scores.

4.2 Qualitative Analysis

4.2.1 T5-Base

Qualitatively, the t5-base model created reasonable predictions, even in specific situations.

Input: Dehumidifier [SEP] A dehumidifier is an air conditioning device which reduces and maintains the level of humidity in the air. This is done usually for health or comfort reasons, or to eliminate musty odor and to prevent the growth of mildew by extracting water from the air. It can be used for household, commercial, or industrial applications. Large dehumidifiers are used in commercial buildings such as indoor ice rinks and swimming pools, as well as manufacturing plants or storage warehouses.

Target: Overview [SEP] Thermal condensation dehumidification [SEP] Absorption/desiccant dehumidification [SEP] Ionic membrane dehumidification [SEP] Condensate [SEP] Maintenance [SEP] Applications

Prediction: Types of dehumidifiers [SEP] Functions of dehumidifiers [SEP] Maintenance of dehumidifiers

Input: Gustatory cortex [SEP] The primary gustatory cortex is a brain structure responsible for the perception of taste. It consists of two substructures: the anterior insula on the insular lobe and the frontal operculum on the inferior frontal gyrus of the frontal lobe. Because of its composition the primary gustatory cortex is sometimes referred to in literature as the AI/FO(Anterior Insula/Frontal Operculum). By using extracellular unit recording techniques, scientists have elucidated that neurons in the AI/FO respond to sweetness, saltiness, bitterness, and sourness, and they code the intensity of the taste stimulus.

Target: Role in the taste pathway [SEP] Functionality and stimulation [SEP] Chemosensory neurons [SEP] Tastant concentration-dependent neuronal activity [SEP] Responsiveness to changes in concentration [SEP] Coherence and interaction of neurons during tasting [SEP] Taste familiarity

Prediction: Structure [SEP] Function [SEP] Clinical significance

Though the model has reasonable predictions, it is still prone to token repetitions, particularly with samples with longer abstracts or more unique topics. In cases with longer abstracts, a larger segment was truncated, losing relevant context for the model. With unique topics, such as drinking water, it would be difficult for the model to have enough context to create relevant sections, this is compounded with the previous abstract truncation problem.

4.2.2 T5-Small

As we can see in the example below, increasing the maximum sequence length from 128 to 256 allows the model to repeat more tokens or even sequence of tokens. In addition, it worth noting that, in general, models using Greedy and Beam search generated shorter sequences than model using Top-p sampling as decoding approach. Also, the model that used Beam search repeats words more frequently than the others. The model that produces the most diverse sequences is the one the uses Top-sampling as decoding method, this shows the efficiency of Top-p sampling in generating explore the diversity.

Input: The alternative sector is a highly profitable industry with a strong lobby, and faces far less regulation over the use and marketing of unproven treatments. Its marketing often advertises the treatments as being "natural" or "holistic", in comparison to those offered by medical science. Billions of dollars have been spent studying alternative medicine, with few or no positive results. Some of the successful practices are only considered alternative under very specific definitions, such as those which include all physical activity under the umbrella of "alternative medicine".

Target: Appeal [SEP] Definitions and terminology [SEP] Types [SEP] History [SEP] Efficacy [SEP] Criticism, legitimacy and effects [SEP] Placebo effect [SEP] Use and regulation [SEP] Conflicts of interest [SEP] Risks.

Prediction - model with 256 tokens: History [SEP] Alternative medicine in the United States [SEP] Alternative medicine in the United States [SEP] Alternative medicine in the United States [SEP] Alternative medicine in the United States.

Prediction - model with 128 tokens and Greedy Search: History [SEP] Practices [SEP] Criticism.

Prediction - model with 128 tokens and Beam Search: Types of antimicrobial resistance [SEP] Antibiotic resistance by type.

Prediction - model with 128 tokens and Top-p sampling: Characteristics [SEP] Effects [SEP] Propagation [SEP] Clinical development [SEP] Practices [SEP] Legal status of alternative medicine [SEP] Research.

5 Conclusion

In this work we explored seq2seq models based on T5 to automatically generate section title for pages from Wikipedia. This is the first step to develop search engines able to generate a new page in order to answer the user's question, summarizing the most relevant information that is originally dispersed throughout the Web. Our experiments shows that T5-Base model can better learn to generate the sections, but it is a large model. In contrast, T5-Small has results similar to T5-Base while having considerably less parameters, hence it is a better option in a limited memory context. Still, we showed that the Greedy Search method for decoding has the best performance, followed by Top-p sampling and Beam Search, respectively. Also, Greedy Search and Beam Search tends to repeat words and produce shorter sequences, while Top-p sampling generates more diverse texts.

6 Future Work

For future work, we intend to experiment other generative models and test other hyperparameters, specially those related to decoding methods, to improve the quality of the generated texts.

References

[1] Laura Dietz, Ben Gamari, and Jeff Dalton. Trec car 2.1: A data set for complex answer retrieval. 2018.

[2] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.

[3] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics.

[4] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[5] Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search for improved description of complex scenes, 2018.