Introduction

Business -> Challenge

Learning summaries of text is a fundamental problem in natural language processing, and of special interest in the context of Large Language Models (LLMs).

Why a challenge? -> Idea

Current approaches that summarize text work in two different ways. The first one is to make use of pretrained embeddings, such as BERT or Word2VEC (Devlin et al. 2018; Mikolov et al. 2013), to apply them to the tokens in the text, and then aggregate these embeddings to a single vector (Shen et al. 2018). The second, more recent, approach is to use generative Large Language Models (LLM) to generate a summary of a text through prompt engineering. Here, we see a gap between these two approaches: While the former can deliver a deterministic numeric summary of the text, it is not clear how to use these summaries for the generation of new text. The latter, on the other hand, can generate new text by prompting for similar writing, but the summary itself is not deterministic (i.e. when re-running the summary prompt, we are not guaranteed the same summary). In our research we aim to bridge this gap by finding a summary embedding that is both deterministic and can be used to generate new text.

Prompt engineering can deliver astonishing results and illustrates the flexibility of LLMs. However, these strategies are difficult to replicate, as they are prone to uncertainty from the sampling process of an LLM, as well as to changes in the model, which are often managed by third parties. For a more robust use in business and academic applications, we need to develop a more systematic approach to the use of LLMs. One prominent application of LLMs is the engineering of features for other downstream tasks.

Our approach, also hinges on LLMs. While these prompt engineering approaches aim at summarizing text with a written summary that is shorter than the original text, we propose a novel approach, which reverse engineers these summaries. Rather than prompting the LLM to write a summary of a given text (the "target sequence"), we perform an optimization to find the input embedding to this LLM, which maximizes the likelihood for the LLM to generate the target sequence.

There are ample use cases and possible empirical investigations for these summary embeddings. In the following, we want to highlight applications both from the perspective of AI development and a domain-specific application in marketing.

This novel training framework can be an important stepping stone towards making generative language models more interpretable and safe. If we can reverse engineer the generation of a specific output sequence, we might be able to perform a sensitivity analysis. Similar to adversarial attacks in computer vision, we can investigate how small changes in the input embedding affect the output sequence. Additionally, there is a potential to learn about the meaning behind different dimensions of the input embedding, as we can investigate how

different elements of the input embedding affect the generated text. An interesting open-source implementation of this could be to build uppon the PyTorch Captum framework, as especially the gradient based methods of this toolbox seem adquate for our problem. Notice that our approach is not limited to a specific domain or Language Model. While we aim to research these summary embeddings in a marketing context, we believe that the use of these embeddings can be extended to other domains in text data, such as pieces of source code, analysis of human feedback, or medical records. It will also be interesting to research, how these summary embedding can be brought to recent developments in the field of multimodal learning, where we aim to learn from both text and images, e.g. to analyze the alignment of text and image for fake news detection (e.g. Uppada, Patel, and B. 2023; Liu et al. 2023).

In the domain of marketing, we envision two projects with these summary embeddings. First, we want to research how marketers design-choices of product claims relates to consumer preferences. For example, a marketer might decide between making a more factual or a more emotional claim. While there are theoretical motivations for these types of decisions, it would allow us to quantitatively assess these styles of claims and to learn how consumer preferences relate to them. An extension of this research could lie in the relation of human marketers and the help of AI, namely LLMs, in the creation of these claims. Investigating how AI and human generated claims that are supposed to fulfill a certain characteristic, such as being more emotional, are actually perceived as such by consumers, could deliver novel insights. Second, in a more field-based setting, we want to investigate how we can use these summary embeddings to adapt online descriptions to the preferences of the consumer. While there are techniques based on Reinforcement Learning, which is used to adapt the website to users' tastes, already in use, these types of algorithms cannot make adaptations "out-of-sample", i.e. they cannot generate new content that is not already pre-defined. We believe that the use of summary embeddings could allow us to generate adaptive text-based content, such as product descriptions or article headlines, that are not part of a predefined set, allowing for a true form of personalization.

Building blocks -> Approach

We aim to link these summary embeddings to consumer preferences, and plan to use these data to inform the generation of new claims for specific contexts.

One way how such a generation could work, is by averaging the embeddings of a set of claims, and then using this average embedding to generate a new claim.

A more sophisticated approach would lie in the use of Multi-Armed Bandits, where we set the arms to be certain elements of the embedding vector. By creating a stack of MAB and LLM, we could give the MAB the ability to generate new claims and to learn about the preferences of the consumer for these new claims. In this, our summary embeddings play a key role, as they allow us to generate new claims in a deterministic fashion, which in-turn enables us to learn about the preferences of the consumer for these new claims.

We could use Combinatorial Bandits to learn about how to create linear combinations of existing embeddings, such that the generated new claim is more appealing to the consumer.

Another empirical applications could lie in learning about consumer preferences for headlines of news articles or posts, and to use these data to inform the generation of new headlines.

We believe this project fits the mission of Amazon Science's Foundation Model Development call for proposals. As it investigates a novel training-framework, which aims to find embeddings that make the LLM generate a pre-determined text, it fits into Theme 2: 'Reducing the sensitivity to tweaks to the input prompt'.

Prior Work

To summarize the information contained in a piece of text, we want to highlight the work on word2vec (Mikolov et al. 2013), GloVe (Pennington, Socher, and Manning 2014), BERT (Devlin et al. 2018), and SBERT (Reimers and Gurevych 2019). The first two word representations are based on the idea that words derive the meaning from the company that they keep, while the BERT and SBERT embeddings are rooted in the use of Transformers (see e.g. Vaswani et al. 2017). Aggregating these word embeddings for sentences can serve as a way to capture the information content in a sentence (see Shen et al. 2018). With respect to the task of computing the similarity between sentences, Reimers and Gurevych (2019) show that their adaptation of the BERT algorithm, SBERT, outperforms other methods with respect to this task and does a good job at serving as a sentence embedding in classification tasks.

Methods

Problem Formulation

To express that we generate tokens with the LLM, $\mathcal{M}$, based on length E input-embedding $\boldsymbol{e}$, we use the notation $\mathcal{M}(\boldsymbol{e})$. To express the number of tokens, k, that we generate, we use the subscript $\mathcal{M}_k(\boldsymbol{e})$. This function call returns an output vector of length V, where V denotes the vocabulary size. We denote the output vector as $\boldsymbol{o}$. Each element o_v of this output vector represents the probability of generating token v as the next token. We predict the next token, by selecting the largest element of $\boldsymbol{o}$. When $k = 1$, we use the previous tokens to predict the next token. When $k > 1$, we append the predicted tokens autoregressively to the input embedding, such that we can predict more than one token. While there are many ways to design this procedure and select tokens along the way, we perform a standard greedy procedure, where we select the token with the highest probability at each step. We denote the probability that $\mathcal{M}$ generates the target sequence, given the input embedding, as $p(t_1, \ldots, t_L|\boldsymbol{e})$. For a given Large Language Model, we define $\boldsymbol{e}^*$,

$$\boldsymbol{e}^* = \arg\max_{\boldsymbol{e}} \sum_{i=1}^{L} \log p(t_1, \ldots, t_L | \boldsymbol{e}),$$

as the input-embedding that maximizes the likelihood of generating a given target sequence, $\{t_i\}_{i=1}^{L}$, where L denotes the length of the target sequence and t_i represents token i. We estimate these summary embeddings $\boldsymbol{e}^*$, by maximizing the conditional likelihood that this embedding generates the target sequence for a given LLM. We use a gradient-based optimization algorithm to maximize this likelihood.

Method in a Nutshell

We aim to find a summary embedding, $\boldsymbol{e}^*$, that maximizes the likelihood of generating a given target sequence, $\{t_i\}_{i=1}^{L}$, with a given LLM, $\mathcal{M}$. We use a gradient-based optimization algorithm to maximize this likelihood. We initialize the summary embedding as the element-wise average of the embedding of the target sequence. We then generate a sequence of length L, with the LLM and the current input embedding. For the resulting output layer, we compute the likelihood that this layer will generate the target sequence. We backpropagate the gradients and adjust the input layer accordingly. We repeat this process until the optimization converges. We want to investigate further improvements to the implementation, such as computing the likelihood contributions of the tokens in a distributed fashion. In Algorithm 1, we summarize the training process.

Algorithm 1 Training of Summary Embeddings

```
Initialize s
i ← 0
while Convergence == False do
    i ← i + 1
    o^(i) ← M(s)
    l^(i) ← Likelihood(o^(i), {t_i}_1^L)
    s^(i+1) ← GradientDecent(s^(i), l^(i))
    Convergence ← CheckConvergence(s^(i+1), s^(i))
end while
```

Expected Results

Goal

Moving forward, we expect to find embeddings that can identify different writing styles across domains. We plan to use these embeddings to generate new text, for example by transfering a

certain writing style from one domain to another, by investigating whether linear combinations of embeddings can be used to generate new, meaningful, text and how this new text relates to the weighted input embeddings. Since this is a novel framework for the training of summary embeddings, we plan to implement this framework as a open-source Python module which is adjusted to the API of the HuggingFace Transformers library library, to ensure ease of use and adoption by other researchers and practitioners, and to scale our research to a large set of LLMs. As mentioned above, we also want to investigate the use of this approach in LLM safety and interpretability, which could also lead to a second open-source implementation, connected to the PyTorch Captum module.

Status quo

At the time of this application, we have implemented an intial version of the training process and have tested it on a small set of target sequences. We have found that the training process is able to find embeddings that make the LLM generate the target sequence. The speed of the optimization depends on the length of the target sequence, size of the LLM, and parameters of the optimization. It seems that even with small LLMs, like GPT-2, the found summary embeddings capture information about the target sequence. For this preliminary test we generated a selection of target sequences through GPT-3.5, prompting it to create advertising slogans for the same product, but with either tangible or more abstract claims. Upon investigating a small selection of the resulting embeddings, we find that they seem to capture inherent information of the target sequence, such as this specific writing style. When visualizing these embeddings, ignoring information on which embedding belong to the tangible and abstract groups, we find a separation between the two groups. On closer inspection, we also find that embeddings of similar target sequences are closer to each other in this embedding space.

Future work

The next steps of this project have different angles.

The first angle is to investigate and understand these summary embeddings better. How do they relate to each other? What types of concepts can we capture with these embeddings? To which degree are they transferable between different versions of the same LLM? How does the captured information compare to existing approaches, such as the use BERT embeddings? Can we learn concepts across domains, such as different writing techniques? A valuable extension here lies in the use of these embeddings together with context information. Can we incorporate information about the author of the target sequence, or the context in which the target sequence was created, into the training process? If so, how does that change the captured information in these embeddings?

As a second angle, we want to investigate how we can use these embeddings to generate new text. A question here is whether linear combinations of these summary embeddings lead to the generation of new, meaningful, text. If this would be the case, then one possible avenue for the generation of new, targeted, text could lie in the use of Multi-Armed Bandits, where we learn how to linearly combine summary embeddings to maximize a reward function, e.g. generating product descriptions to maximize a key performance measure such as click-through rate.

We plan the following timeline:

Novel hardware exploration

Despite working on an efficient implementation of our algorithm and the ability to use pre-trained LLMs, we are currently constrained by the computational resources available to us, which limits the lengths of our feasible target sequences. While we can already train summary embeddings for target sequences of lengths that are useful for applications e.g. in marketing, summarizing full documents has prohibitively high computational costs. Being able to perform these calculations for full "target documents", could lead valuable insights for example when researching political polarization. If these summary embeddings truly pick up on nuances in the writing style, we could use them to investigate how different political groups and news outlets differ in the bias of their writing style. Since, we are interested in using these embeddings for the generation of new text, we might be able to use these deterministic summary to "de-bias" a piece of writing in a generative and deterministic fashion. These types of applications hinge on the availability of computational resources, and the use of the AWS Trainium2 chips could be a valuable asset in this regard.

The higher computational efficiency of the Trainium2 chip also aligns with our view, that we need to be mindful of the environmental impact of our research. We try to keep this impact low by only making use of pre-trained LLMs and only having to perform forward-passes through these. However, we are aware that the repetitive

For applications of these summary embeddings to massive datasets of millions of target tokens, a high degree of parallelization will be necessary to perform this in adequate time. While each separate optimization only requires a sequence of forward-passes through the LLM, running these optimizations in parallel will require a high degree of computational resources. We would be excited to investigate how such a training process could be tailored to the architecture of the Trainium2 hardware.

Currently, we are working with small LLMs such as GPT-2, to keep our training times at bay and computational costs low. Naturally, we are interested in the potential of using state-of-the art LLMs instead, such as e.g. LLaMA2. Not only can these types of models generate more coherent and meaningful text, but their input embeddings are of higher dimensions than the current LLMs we are using. This could potentially lead to more expressive and more detailed summary embeddings.

Open-source intention

This grant would enable us to contribute to the open-source community in two ways. The first way is by open-sourcing our code in a publicly hosted Python module. Ideally, this module would be adjusted to the API of the HuggingFace Transformers library. This would ensure ease of use and adoption by other researchers and practitioners, and would allow us to scale our research to a large set of LLMs. In the context of using these summary embeddings for researching LLM safety and interpretability, an extension to the PyTorch Captum module would prove to be valuable. The second way how we can contribute to the open-source community, is not through code, but by providing access to pre-trained summary embeddings. With sufficient computing resources, we would be able to train these summary embeddings for a large set of target sequences from a public dataset, and make these embeddings publicly available to the research community for benchmarking purposes. One example could be to provide summary embeddings for the Microsoft News Dataset, which serves as a benchmark for news recommendation systems.

Funds needed

- Current implementation needs around 3 minutes per claim, on Google Colab T4 GPU.
- Training of embeddings in parallel
- GPU is the limiting factor

References

Since we are working within the real of LLMs, we aim to incorporate our training framework in the API of the HuggingFace Transformers library. This would ensure ease of use and adoption by other researchers and practitioners, and would allow us to scale our research to a larger set of LLMs.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." https://doi.org/10.48550/ARXIV.1810.04805.

Liu, Peng, Wenhua Qian, Dan Xu, Bingling Ren, and Jinde Cao. 2023. "Multi-Modal Fake News Detection via Bridging the Gap Between Modals." *Entropy* 25 (4): 614. https://doi.org/10.3390/e25040614.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. "Distributed Representations of Words and Phrases and Their Compositionality." *Advances in Neural Information Processing Systems* 26.

Pennington, Jeffrey, Richard Socher, and Christopher Manning. 2014. "GloVe: Global Vectors for Word Representation." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, edited by Alessandro Moschitti, Bo Pang, and Walter Daelemans, 1532–43. Doha, Qatar: Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162.

Reimers, Nils, and Iryna Gurevych. 2019. "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks." https://doi.org/10.48550/ARXIV.1908.10084.

Shen, Dinghan, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. "Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms." https://doi.org/10.48550/ARXIV.1805.09843.

Uppada, Santosh Kumar, Parth Patel, and Sivaselvan B. 2023. "An Image and Text-Based Multimodal Model for Detecting Fake News in OSN's." *Journal of Intelligent Information Systems* 61 (2): 367–93. https://doi.org/10.1007/s10844-022-00764-y.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." *Advances in Neural Information Processing Systems* 30.