# Introduction

The use of LLMs in research and business applications is on the rise.

Prompt engineering can deliver miraculous 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.

Learning summaries of text is a fundamental problem in natural language processing, and of special interest in the context of Large Language Models (LLMs). While current 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 a LLM to write a summary of a given text, we perform an optimization to find the input embedding to this LLM, which maximizes the likelihood for the LLM to generate the given text.

Current approaches that summarize the information in a piece of text work in two different ways. The first, longer established, one is to make use of pretrained embeddings, such as BERT or Word2VEC (REF), to apply them to the tokens in the text and then aggregate these embeddings to a single vector. The second, more recent, approach is to use LLMs to generate a summary of the text. 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, but the summary is not deterministic. 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.

We can understand summaries of text in two different ways. First, we can understand them as a way to compress the information in a piece of text to a single vector or matrix. Second, we can understand them as a “written” summary, which is a piece of text that is shorter than the original text.

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.

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.

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.

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.

There are ample use cases and possible empirical investigations in a marketing context for these summary embeddings. In the following, we want to highlight three different applications, that might deliver novel insights due to the use of these summary embeddings.

First, we want to research how marketers design choices on the creation 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 the preferences of the consumer, already in use, these types of algorithms cannot make adaptations “out-of-sample”, i.e. they cannot generate new content that is not already predefined. 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.

In preliminary tests, we have found that these summary embeddings appear to be able to differentiate different styles of claims in the same domain.

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’.

# Methods

* Novel, “reverse engineering” approach to finding text summaries (find a better word… in an information sense, not in a writing short summary sense). Given a target sequence and a given LLM, we want to find the input embedding to this LLM, which maximizes the likelihood for the LLM to generate the target sequence.
* Training can be done with simply forward passes through the LLM model

## Problem Formulation

For a given Large Language Model, , we want to find the input embedding, , that maximizes the likelihood of generating a given target sequence, , where denotes the length of the target sequence and represents token .

To express that we generate tokens with the LLM, based on length input-embedding , we use the notation . To express the number of tokens, that we generate, we use the subscript . This function call returns an output vector of length , where denotes the vocabulary size. We denote the output vector as . Each element of represents the probability of generating token as the next token. We predict the next token, by selecting the largest element of . Explicitly, $t\_{1} = \argmax \mathcal{M}\_1(\boldsymbol{e})$.

When , we use the previous tokens to predict the next token. When , 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 generates the target sequence, given the input embedding, as .

We estimate these embeddings, 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.

These embeddings have the same dimension as the input embedding of , length .

$$
\boldsymbol{s^{\*}} = \argmax\_{\boldsymbol{s}} \prod\_{i=1}^{L} p(t\_1, \ldots, t\_L | \boldsymbol{s})
$$

$$
\boldsymbol{s^{\*}} = \argmax\_{\boldsymbol{s}} \sum\_{i=1}^{L} \log p(t\_1, \ldots, t\_L | \boldsymbol{s})
$$

$$
\boldsymbol{s^{\*}} = \argmax\_{\boldsymbol{s}} \log p(t\_1, | \boldsymbol{s}) + \log p(t\_2, | \boldsymbol{s}, t\_1) + \ldots + \log p(t\_L, | \boldsymbol{s}, t\_1, \ldots, t\_{L-1})
$$

$$
\boldsymbol{s^{\*}} = \argmax\_{\boldsymbol{s}} \log \mathcal{M}\_1(\boldsymbol{s}) + \log \mathcal{M}\_1(\left [\boldsymbol{s}, \boldsymbol{e}\_1 \right]) + \ldots + \log \mathcal{M}\_1(\left [\boldsymbol{s}, \boldsymbol{e}\_1, \ldots, \boldsymbol{e}\_{L-1} \right])
$$

$$
\boldsymbol{s^{\*}} = \argmax\_{\boldsymbol{s}} \sum\_{i=1}^{L} \log \mathcal{M}\_L (\boldsymbol{s})
$$

We can make this expression more explicit, by using the LLM in our notation. For example

In the training process, we execute the following algorithm. First, we initialize the input embedding, . Second, we generate a sequence of length , with the LLM and the current input embedding.

Currently, we initialize the summary embedding as the element-wise average of the embedding of the target sequence. This speeds up the computation, compared to a randomly intialized embedding. We want to investigate further improvements to the implementation, such as computing the likelihood contributions of the tokens in a distributed fashion.

## Method in a Nutshell

# Expectued Results

## Goal

Preliminary results show that we can find embeddings that generate the target sequence with a high likelihood. Upon investigating a small selection of the resulting embeddings, we find that they seem to capture inherent information of the target sequence, such as a specific writing style. When visualizing these embeddings, with the help of PCA, in a 2D space, we find that embedding of similar slogans tend to be close to each other.

We want to investigate different variations of the training process, such as the use of a smaller neural network to generate the embeddings, allowing it to also learn about contextual variables, such as the author of the target sequence.

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. We also want to investigate whether linear combinations of embeddings can be used to generate new, meaningful, text and how this new text relates to the weighted input embeddings.

## Status quo

## Future work

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.