working title

Finn-Ole Höner (657110)

June 15, 2024

Introduction

> *"Prompt engineering is the art of communicating eloquently to an AI."—-Greg Brockman, President OpenAI*

Generative Large Language Models (LLM) are powerful, but unreliable tools for generating text. A study by the NGO "Algorithm Watch", which is partially funded by the European Union, finds that Microsoft's chatbot Bing Chat, delivers wrong information in a third of its replies to election related questions, posing a risk to exacerbate desinformation.

Prompt engineering defines the task of finding the right prompt to generate a desired output. Typically, this is an interactive and iterative process between human and LLM. There are certain writing techniques that improve the quality of the LLM's response, such as providing examples for what the desired output should look like. There are various blog posts and books (e.g. Phoenix and Taylor 2024) about how to write good prompts, in fact businesses are hiring "Prompt Engineers" even on dedicated job boards. Despite these, supposedly low-barrier, resources, end users without AI knowledge still struggle with prompt engineering (Zamfirescu-Pereira et al. 2023). Not only can prompt engineering pose a security risk, but the process is also fuzzy, hard to replicate, and might still lead to unpredictable behaviors of the model. These issues pose risks for organizations, as the obscurity of manual prompt engineering can lead to loss of know-how, and the unpredictiability of the model can lead to lawsuits, damages to brands, or missed opportunities.

Problems of reproduceability, transparency, and documentation, become critical for organizations with the advent of the EU AI Act. The act requires for "High risk applications", that the AI model features, e.g. "adequate mitigation systems", "traceability of results", and, "robustness". High risk applications are not common and important for society, they include applications e.g. in education, employment, and public services.

We propose a new type of numeric document summary, which captures all the information contained in the document, as it can be used to recreate the document itself. This document summary comes in the form of a numeric vector, which we call the summary embedding. To obtain this summary embedding, we maximize the likelihood of generating the document with a Large Language Model (LLM). In this sense, these summary embeddings are optimal. We show, that these embeddings capture inherent information about the document, can be used for classification, and generation of new documents. There are only practical requirements for estimating these summary embeddings, namely that backpropagation of gradients is possible from the likelihood to the input of the LLM.

Beyond the methodological contribution, we show a marketing application of these summary embeddings and make a step towards using Generative AI in market research and design of the marketing-mix. We find that we can use these embeddings to capture design motivations of marketers behind advertising claims, and establish a measure for linguistic uniqueness of an advertising claim. Suprisingly, we find that this linguistic uniqueness is negatively correlated with the perceived uniqueness of an advertising claim by consumers. We explore these linguistic features and find that claims with certain words and

their synonyms, cluster together, e.g. claims that emphasize the taste of a yoghurt drink or present it as a breakfast. Furthermore, we show that claims that live in certain areas of a low dimensional representation of the summary embeddings, are more likely to be rated favorably by consumers, but that small differences in wording can lead to relatively large differences in evaluation. This suggests a two-step approach to designing a market research study for advertising claims, where the first stage should focus on a wide exploration of the design space, while the second stage should focus on a fine-grained evaluation of the most promising area in this design space. We end with a sketch on how these summary embeddings can be used in an AI augmented design process, similar to applications in image data by Burnap, Hauser, and Timoshenko (2023) and Ludwig and Mullainathan (2023).

Relevant Literature

Generative Large Language Models (LLM), such as ChatGPT (Radford et al. 2018), generate text based on textual inputs, so called prompts. Internally, the model translates the prompt first into tokens, integer codes representing certain words, symbols and syllabuls, and then into a lower dimensional vector representation of the text itself, which we call the input embedding. This input embedding is then passed through a neural network that yields a probability distribution over the next token in the sequence. To generate text, the model makes a draw from this distribution, and attached this generated token at the end of the prompt, repeating this process until it predicts the next token to be the end-of-text token (eos), i.e. until it predicts that the text ends. LLMs are pre-trained on large text corpora, such as the Common Crawl which contains over 250 billion pages of text. Their architecture is rooted in the attention mechanism. The attention mechanism models interactions in the text, such as negations, synonyms, or grammatical structures, across a long sequence of text. Vaswani et al. (2017) implement this mechanism in the transformer block, which is a modular building block for the neural network underlying an LLM. The combination of these transformer blocks, giant pre-training data, and huge computing resources, lead to LLMs that have "emergent capabilities". These are capabilities, which were not explicitly trained for by the developers of the model, such as the ability of a model to translate between languages, pass the BAR exam, or write textual summaries of documents (see Wei et al., n.d.; Lu et al. 2023). Unidirectional LLMs, such as GPT or LLaMa (Radford et al. 2018; "Meta Llama 3 — Llama.meta.com," n.d.), are built for text-generation. They predict the next token in a sequence of tokens autoregressively, therby, these models only work from left-to-right (or right-to-left). Bidirectional LLMs, such as BERT (Devlin et al. 2018), are built for representing text. They predict a token given the context around this token, and can hence read the text both from left-to-right and right-to-left. Both types of models share two note-worthy components, that account for a large share of their power. One, developers pre-train both types of models on large bodies of text, to learn about the structure of language itself. Two, by using a transformer architecture (Vaswani et al. 2017), which enables these language models to learn dependencies between words, such as negations (short-range dependency), or document structures, e.g. salutations in letters (long-range dependency).

In the following, we compare our summary embeddings with three established methods to summarize documents: The Bidirectional Encoder Representations from Transformers (BERT) model's `[CLS]` token (Devlin et al. 2018), pooled word embeddings (Shen et al. 2018), and prompt engineering approaches (@ Huang and Chen, n.d.).

The BERT model is a transformer (see Vaswani et al. 2017) based model to represent text. Its developers have pre-trained BERT on a large corpus of text, using two different self-supervised learning objectives. The first objective is the so called "Masked Language Model" task. Here, the researchers replace a random selection of tokens in the text by a mask token, and the goal of the BERT model is to predict which token is underlying this mask. In order to learn about the information captures in a sentence, the researchers also pose the "Next Sentence Prediction" task to the model. For this training objective, the BERT model receives a pair of two sentences and needs to predict whether the second sentence follows the first one in a text. This task trains the `[CLS]` token: As it is pre-pended before every such sentence pair, it learns a representation of the information contained in the focal sentence. In practice, the output for a `[CLS]` token serves as a powerful feature e.g. for sentence classification. BERT embeddings are versatile, but can also be fine-tuned to a specific task such as text summarization. However, such a fine-tuning requires the user to

compile a dataset of text-summary and text pairs, that are representative of their application. These BERT representations are only descriptive and cannot be used for the generation of new text (see Devlin et al. 2018). There exist expansions to BERT, such as RoBERTa by Yinhan Liu et al. (2019), which improve the optimization procedure.

Shen et al. (2018) points to the value of using pooled word embeddings to represent documents. The idea behind a pooled word embedding is, that we can represent a sequence of tokens by aggregating the word-embeddings of these tokens. There are different methods we can use for this aggregation, some examples are the use of taking the average across the tokens (mean-pooling) or taking the maximum value across the tokens (max-pooling). There are various types of word embeddings that we could use for the pooling, often, these are non-attention word embeddings, such as Word2Vec (Tomáš Mikolov, Mikolov, Ilya Sutskever, et al. 2013; Tomáš Mikolov, Mikolov, Kai Chen, et al. 2013) or Global Vectors (GLoVe) (Jeffrey Pennington et al. 2014). These pooled word embeddings, cannot be used for text generation and perform worse than BERT embeddings in language tasks, as they do not employ the attention mechanism and loose information in the aggregation step (see e.g. Onan 2023).

We can also summarize documents by passing these to an LLM and prompting the model to write a summary for us (e.g. Huang and Chen, n.d.; Chakraborty and Pakray 2024). However, these summaries are textual, not deterministic, and their quality depends on the formulation of the prompt. For example, Liu et al. (2023) find that answers of LLMs are better, when we place relevant information at the beginning or end of the prompt, while inserting emotions into prompts also improves the quality of the LLMs response, in some cases even doubling its performance (see Li et al. 2023). Another issue is that these summaries can vary in length, and one needs to specify how detailed a summary should be. The difference in length between the summary and focal document, also makes it challenging to evaluate how much of the information of the focal document is captured by the summary (Chakraborty and Pakray 2024). If these summaries are of a high quality, i.e. represent the information in the focal document well, then they can be useful when we want to share a written summary of a document. However, for many data science and automation tasks, we need a reliable numeric representation of the document, which is the focus of this paper.

In this work, we propose a method that optimizes the input prompt to an LLM, such that we obtain a desired outcome. We call these input prompts "document summaries", as the perfectly replicate a focal document. These input embeddings capture all information that is contained in a document, because we can re-generate the focal document just based on this summary embedding. These document summaries are deterministic, because we obtain them through maximum likelihood estimation and live in continuous space. We can evaluate the fit of such a document summary, as the likelihood indicates its fit directly.

Table 1: Overview of existing methods to summarize documents.

Method	Origin	Reference	Deterministic?	Generation?	Type?
BERT `[CLS]` token	Next-sentence prediction task	Devlin et al. (2018)	✓	×	Numeric
Pooled Word Embeddings	Aggregation of token information	Shen et al. (2018)	✓	×	Numeric
Prompt engineering	Emergent capability of LLM	CITE	×	×	Textual
Reverse engineered document summaries	Maximize the likelihood to re-generate the focal document	*Proposed method*	✓	✓	Numeric

Methodology

- For generation of sequences, the LLM autoregressively predicts the next token, given the previous tokens. At each step in the generation, a selection of the next token needs to take place, for which

there exist various procedures (CITE). As we are optimizing with respect to the joint likelihood of the target sequence, we also need to generate with a procedure that generates this most likely sequence. Finding this sequence from the generation tree is a combinatorical problem and computationally not feasible. Hence, we use the beam-search heuristic to make our generation. Rather than only tracking the most likely token at each generation step, beam search also tracks the n-most likely tokens. The more beams we search, the more likely we are to find a sequence with a higher likelihood, then we would have found otherwise ("How to Generate Text: Using Different Decoding Methods for Language Generation with Transformers — Huggingface.co," n.d.).

Each document, in our case an advertising claim, d consists of the tokens $t_1^d, \ldots, t_T^(d)$.

The probability to generate the sequence $t_1, \ldots, t_T$ with the LLM when using the embedding $\boldsymbol{s}$ as an input is $p(t_1, \ldots, t_T \mid \boldsymbol{s})$. We can split this probability into conditional probabilities due to the autoregressive architecture of the LLM and because we observe the target sequence. This provides large computational gains, as we can calculate these parts in parallel. Hence, we define the log-likelihood of the summary embedding for the target sequence as

$$\mathcal{L}(\boldsymbol{s} \mid t_1, \ldots, t_T) = \log p(t_1|\boldsymbol{s}) + \log p(t_2|t_1, \boldsymbol{s}) + \ldots + \log p(t_T|t_1, \ldots, t_{T-1}, \boldsymbol{s}),$$

and find the optimal summary embedding as

$$\mathbf{s}^* = \arg\max_{\boldsymbol{s}} \mathcal{L}(\boldsymbol{s} \mid t_1, \ldots, t_T).$$

We summarize the training process for the single summary embeddings in Algorithm 1. The summary embedding is a vector of length E, which is the same dimension as the dimension of the LLM's input embedding.

Algorithm 1 Training of Single Summary Embeddings

$i \leftarrow 0$
$\epsilon \leftarrow 0.01$
$\boldsymbol{s} \leftarrow \text{Initialization}(\cdot)$
while $True$ **do**
  $l^{(i)} \leftarrow \mathcal{L}(\boldsymbol{s} \mid t_1, \ldots, t_T)$
  $\nabla_{\boldsymbol{s}}^{(i)} l \leftarrow \text{ComputeGradient}\left(l^{(i)}\right)$
  $\boldsymbol{s}^{(i+1)} l \leftarrow \text{Optimizer}\left(\boldsymbol{s}^{(i)}, \nabla_{\boldsymbol{s}}^{(i)} l\right)$
  $i \leftarrow i + 1$
  if $l^{(i)} < \epsilon$ **then**
    break
  end if
end while

As we are interested in finding a low-dimensional sub-space, in which we can represent the D advertising claims, we use a factor model and estimate the summary embeddings for all advertising claims jointly, yielding the $D \times E$ matrix of summary embeddings $\mathbf{S}$. We restrict the degrees of freedom to F hidden factors. To implement this factor model, we create an Autoencoder with one hidden layer, that is fully connected with linear activation functions (Goodfellow, Bengio, and Courville 2016). The input data to this autoencoder is an identity matrix of dimension D. We illustrate this neural network representation of the factor model in Figure 1.

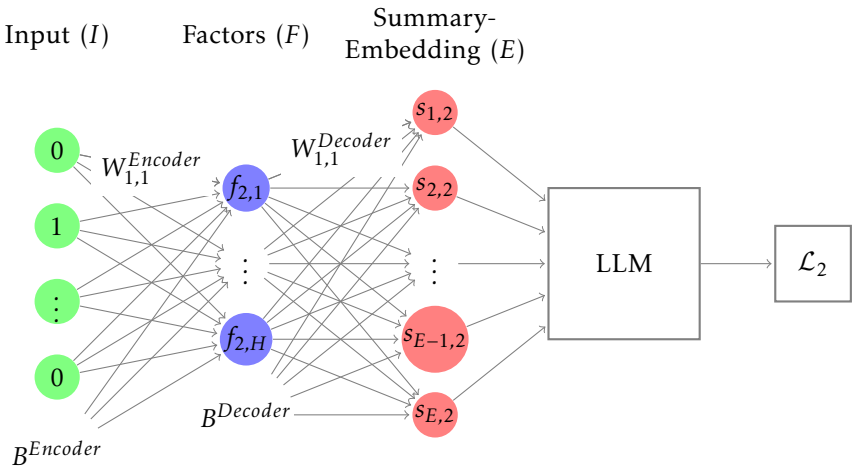


Figure 1: Factor model in Neural Network form, example for document 2

We denote the matrix of weights of the encoder and decoder layers by $\mathbf{W}^{Encoder}$ and $\mathbf{W}^{Decoder}$. These matrices have the dimensions $D \times F$ and $F \times E$ respectively. We also define the vectors of biases for the encoder and decoder by $\boldsymbol{b}^{Encoder}$ (length F) and $\boldsymbol{b}^{Decoder}$ (length E), and stack them into matrices by taking the Kronecker product with a length D vector of ones, $\mathbf{1}_D$, to obtain the biases in matrix form as $\mathbf{B}^{Encoder} = \mathbf{1}_D \otimes \boldsymbol{b}^{Encoder}$ and $\mathbf{B}^{Decoder} = \mathbf{1}_F \otimes \boldsymbol{b}^{Decoder}$. For notational convenience, we denote this factor model by $\text{Factor}_{\Omega_{Factor}}(\mathbb{I}_D) : \{0,1\}^{D \times D} \to \mathbb{R}^E$, with parameters Ω_{Factor}, where $\Omega_{Factor} = \{\mathbf{W}^{Encoder}, \mathbf{W}^{Decoder}, \mathbf{B}^{Encoder}, \mathbf{B}^{Decoder}\}$.

When we define the weights and biases in this form, we can write the factor representation of the input as

$$\mathbf{F} = \mathbf{W}^{Encoder} + \mathbf{B}^{Encoder},$$

thereby the matrix of summary embeddings for all claims becomes

$$\mathbf{S} = \left(\mathbf{W}^{Encoder} + \mathbf{B}^{Encoder}\right)\mathbf{W}^{Decoder} + \mathbf{B}^{Decoder}.$$

When we estimate the matrix of summary embeddings $\mathbf{S}$, we use the joint log-likelihood to re-generate all documents in the dataset, $\mathcal{L}_D = \sum_{d=1}^{D} \mathcal{L}_d$, where $\mathcal{L}_d$ is the log-likelihood to re-generate document d with summary embedding $\mathbf{S}$. We estimate the summary embeddings by maximizing this joint log-likelihood with respect to the parameters of the factor model:

$$\mathbf{S}^* = \underset{\mathbf{W}^{Encoder}, \mathbf{W}^{Decoder}, \mathbf{B}^{Encoder}, \mathbf{B}^{Decoder}}{\arg\max} \mathcal{L}_D(\mathbf{S}).$$

We summarize the training process for finding the sub-space representation in Algorithm 2.

Diagnostics

- Other methods as benchmarks
  - Word2Vec
  - BERT

To analyze our results, we will use a suite of diagnostic tools, which we will explain here briefly.

Algorithm 2 Training of Summary Embeddings based on factor model

$i \leftarrow 0$
$\epsilon \leftarrow 0.01$
$\mathbf{W}_{(i)}^{Encoder}$, $\mathbf{W}_{(i)}^{Decoder}$, $\mathbf{B}_{(i)}^{Encoder}$, $\mathbf{B}_{(i)}^{Decoder} \leftarrow \text{Initialization}(\cdot)$
while $True$ **do**
  $\mathbf{S} \leftarrow \left(\mathbf{W}_{(i)}^{Encoder} + \mathbf{B}_{(i)}^{Encoder}\right)\mathbf{W}_{(i)}^{Decoder} + \mathbf{B}_{(i)}^{Decoder}$
  $l_{(i)} \leftarrow \mathcal{L}_D(\mathbf{S})$
  $\nabla_{(i)} \leftarrow \text{ComputeGradient}\left(l^{(i)}\right)$
  $\mathbf{W}_{(i+1)}^{Encoder}$, $\mathbf{W}_{(i+1)}^{Decoder}$, $\mathbf{B}_{(i+1)}^{Encoder}$, $\mathbf{B}_{(i+1)}^{Decoder} \leftarrow \text{Optimizer}\left(\mathbf{W}_{(i)}^{Encoder}, \mathbf{W}_{(i)}^{Decoder}, \mathbf{B}_{(i)}^{Encoder}, \mathbf{B}_{(i)}^{Decoder}, \nabla_{(i)}\right)$
  $i \leftarrow i + 1$
  if $l_{(i)} < \epsilon$ **then**
    break
  end if
end while

- Correlation matrix
  - First, we de-mean each embedding dimension by subtracting the mean of this embedding dimension calculated across observations. The motivation for this is, to take out parts of the embedding that are due to the domain of the advertising claims.
  - Correlation matrices across the claims and across the embedding dimensions: When analyzing the summary embeddings, we form correlations across the advertising claims and correlations across the embedding dimensions. The former is the same as a correlation matrix of a data matrix with as many rows as embedding dimensions and as many columns as the number of observations, while the latter is the correlation matrix of a data matrix with as many rows as observations and as many columns as embedding dimensions.
- Leave-one-out regression (LOO)
  - We also perform leave-one-out regression, here we regress a focal embedding onto the average embedding from the one class and the average embedding from the other class, and an intercept. We repeat this for all observations. Since all embeddings are de-meaned, we expect an average intercept of zero for all regressions. We also expect that the estimates for the focal claim's group are positive, while the estimates for the other group should be negative.
- Principal Component Analysis (PCA)
  - PCA is a dimensionality reduction technique, which is rooted in the singular value decomposition. Intuitively, it projects the data onto orthogonal axes in a way that each axis captures as much variance as possible. We select the first two principal components, as this allows us to visualize our data.
  - Eigenvalue? Scree plot?

Data

> **Prompt:** *I work for a marketing agency. I want to market a haircare product and need some claims for this.* ***I want you to emphasize how shiny the haircare product consumers' hair makes****. I need claims that differ in style with respect to how tangible the claims are. Make these claims brief. Can you give me 10 claims that are rather tangible and 10 claims that are rather intangible?*

Figure 2: Prompt to generate the benchmark advertising claims

As as robustness check, we repeated this task for different attributes (instead of shiny, healthiness and colorfulness of hair), and for a different product category (surface cleaners). In total, we obtain 80 advertising claims, 20 for each of these settings. Half of these claims are supposed to be tangible and half intangible.

We obtain data from a market research company, that includes advertising claims for different FMCG. In our application, we focus on dairy and hair shampoo products. These data, stem from different variations of choice-based conjoint studies (Eggers et al. 2022), which we cannot further disclose due to confidentiality agreements. The data includes choice-experiments aggregated at the level of the advertising claims, hence we do not have responses of individuals. These measurements of consumer-preferences are on multiple dimensions, such as relevance, uniqueness of a claim, brand fit, or an overarching rating of the claim. The advertising claims were designed for markets in different countries and are in english (US, UK) or translated into english (others). The date of the study is also included in the data. We also have information on the marketers motivation behind the design of an advertising claims, such as whether she designed the claims to pronounce the health benefits of a yoghurt, or the taste yoghurt, as well as the brand of the product.

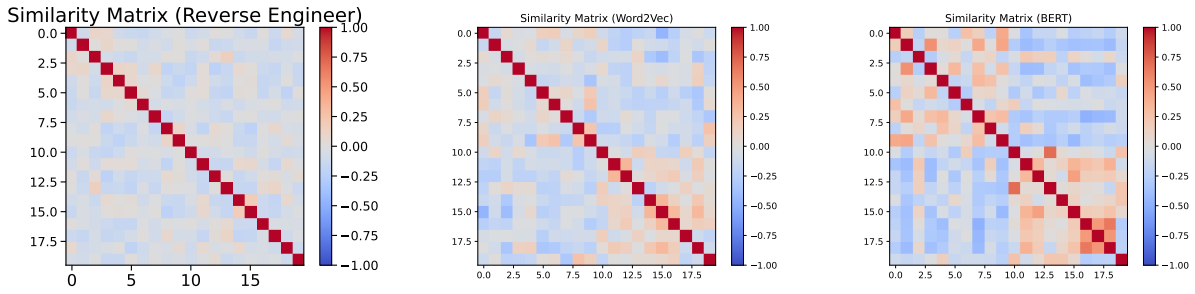
- Descriptives of the data
- Prepocessing of the data

Results

Single Embeddings

- Reproduce the respective claims

Figure 3



(a) RE (b) W2V (c) BERT

Figure 3: Correlation matrices along the claims.

- does not work to capture the two classes: Overfitting?
- does regularization help?
- What about interpolation

Factor Representation

For our factor model, we use two factors. The intuition for this is, that we want to distinguish two classes of tokens.

- Performed suite of tests of generated data
  - haircare shiny
  - haircare healthy
  - haircare colorful

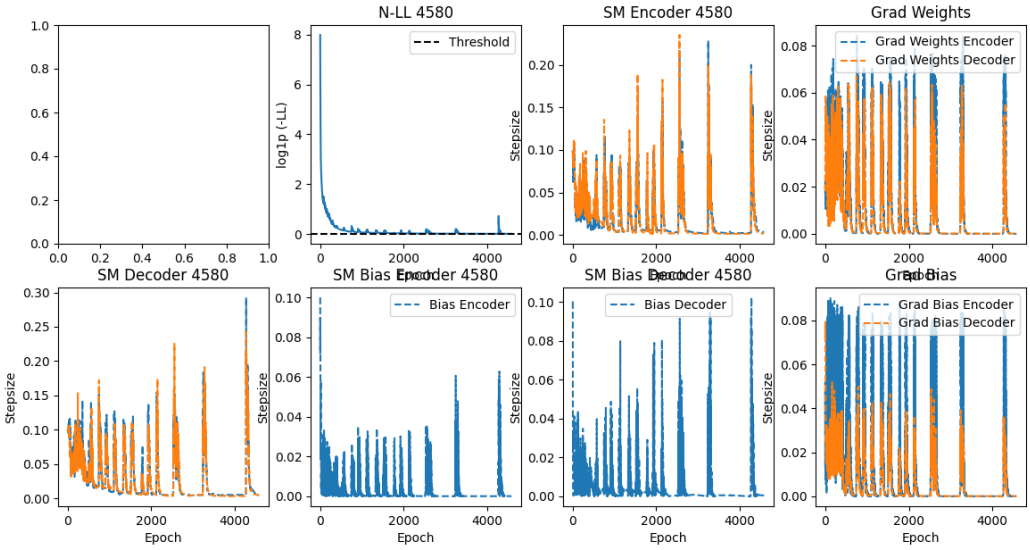


Figure 4: Training of embeddings

- computation
- dealing with spikes

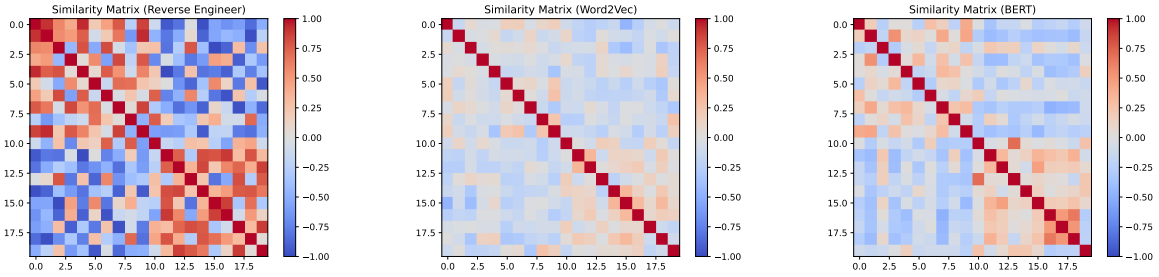
[ ] Analysis of embeddings; Plot correlation of claims and dims, PCA

If our summary embeddings are able to capture relevant information in the advertising claims, then we expect to be able to separate between the tangible and intangible advertising claims.

For this, we demean the embeddings across the embedding dimensions such that the only variation left should be due to the claims being tangible or intangible.

When visualizing these correlations, we expect the first quartile and the third quartile to contain positive correlations, while the second and the fourth quartile should contain negative correlations. In Figure 5 we present these correlation matrices for our Summary embeddings, Word2Vec embeddings, and BERT embeddings. We see this pattern for all three methods, indicating that all three methods are able to capture this part of information in the advertising claim. However, the correlations are large (in absolute value) for

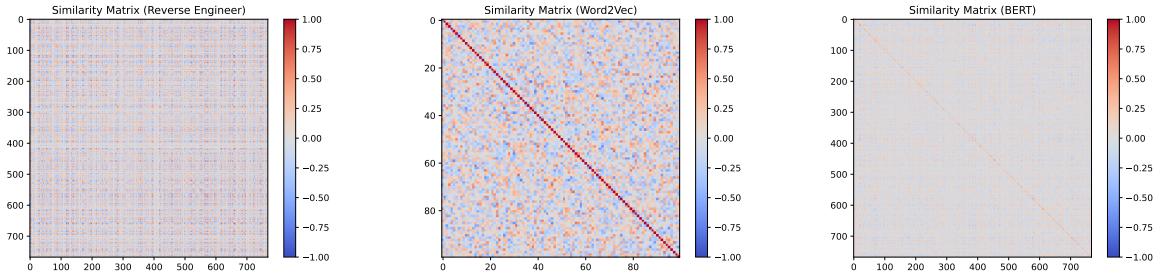
the summary embeddings. This is a first indication, that our factor structure captures this difference in the advertising claims.



(a) RE (b) W2V (c) BERT

Figure 5: Correlation matrices along the claims.

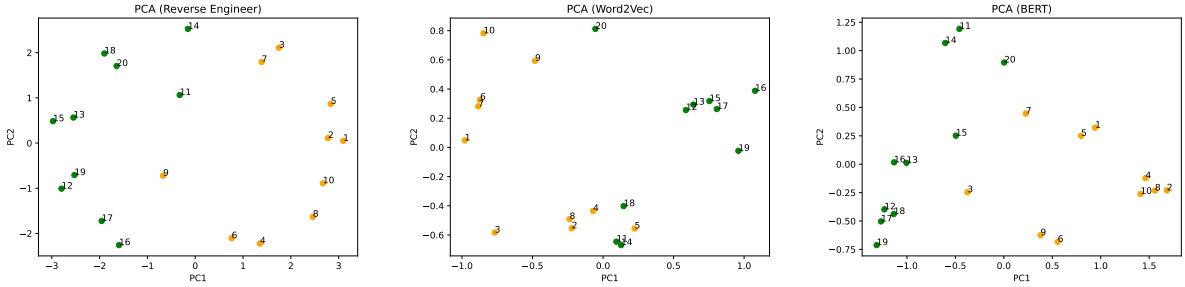
Figure 6 shows correlation matrices across the embedding dimensions, i.e. each cell shows the correlation between two embedding dimensions calculated across the 20 advertising claims. When comparing our summary embedding and the BERT based embeddings, we can see a stronger grid pattern with more correlations that are removed from zero. this is something we would anticipate for our factor model, as we restricted the degrees of freedom of the elements in the embedding vector to 2. Hence, many of the dimensions should be linearly dependent on each other. In contrast, for the BERT embedding most dimensions are barely correlated with each other.



(a) RE (b) W2V (c) BERT

Figure 6: Correlation matrices along the embedding dimensions.

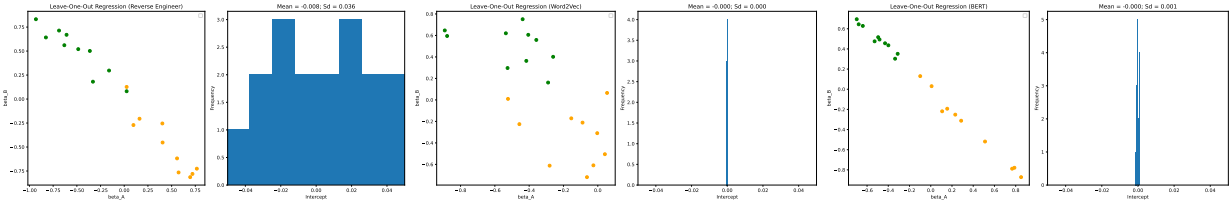
We perform Principal-Component Analysis on the embedding matrices and visualize the first two principals components in Figure 7. We color the data points by their respective class. We see that we can separate these two classes linearly for all three embeddings.



(a) RE (b) W2V (c) BERT

Figure 7: First two principal components of the embedding matrix, colored by class.

Figure 8 shows the results of the leave-one-out regression. For the summary embeddings, the variance of the intercept is substantially larger than for the other embeddings.



(a) RE (b) W2V (c) BERT

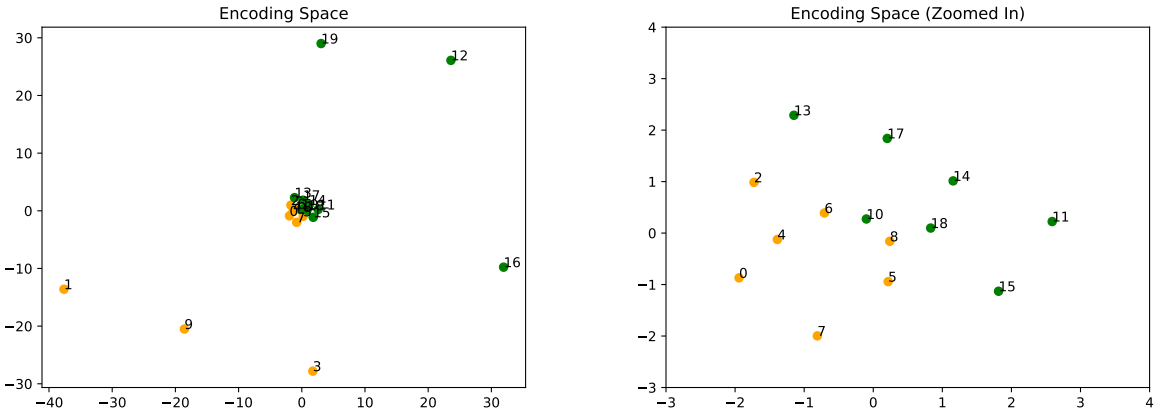
Figure 8: Reuslts of leave-one-out regression, colored by class.

- GPT-2
- BERT
- Word2Vec
- RE

[ ] Encoding dimension plot

Figure 9 shows the low dimensional sub-space in which we located the advertising claims. It appears that we pick up the class with the two hidden factor, however, the class is not directly aligned with one of these axes. Many of the points are close to zero, with a few of the claims being far removed from the other points, e.g. claims 1, 9, and 3 for the tangible, and claims 12, 16, and 19 for the intangible claims.

Table 2



(a) All points

(b) Zoom-in

Figure 9: Encoding space of the advertising claims. Numbered points are locations of claims that are part of the training data, colored dots are points where we generate one of the claims that are part of the training data, and red crosses are locations where we generate a claim that ends with the eos-token, but is not part of the training data. Whitespace marks coordinates where we generate a string that does not contain an eos token.

Table 2: Tangible (T) and Intangible (I) advertising claims for hair shampoo products.

Number	Claim
0 (T)	Experience 50% more visible shine after just one use.
1 (T)	Formulated with light-reflecting technology for a glossy finish.
2 (T)	Transform dull strands into radiant, luminous locks.
3 (T)	Infused with nourishing oils that enhance natural shine.
4 (T)	See instant brilliance with our advanced shine-boosting formula.
5 (T)	Locks in moisture to amplify hair's natural luster.
6 (T)	Achieve salon-quality shine without leaving home.
7 (T)	Visible reduction in dullness, replaced with stunning shine.
8 (T)	Say goodbye to lackluster hair, hello to mirror-like shine.
9 (T)	Clinically proven to enhance shine by up to 70%.
10 (I)	Elevate your confidence with hair that gleams under any light.
11 (I)	Embrace the allure of luminous hair that turns heads.
12 (I)	Unleash the power of radiant hair that speaks volumes.
13 (I)	Transform your look with hair that exudes brilliance.
14 (I)	Feel the difference of hair that shines with vitality and health.
15 (I)	Rediscover the joy of hair that beams with inner vibrancy.
16 (I)	Indulge in the luxury of hair that shimmers with elegance.
17 (I)	Step into the spotlight with hair that radiates beauty.
18 (I)	Experience the magic of hair that dazzles with every movement.
19 (I)	Unlock the secret to hair that shines from within, reflecting your inner glow.

- labels, accuracy
- clustering
- generation of new claims
  - exploring the space
  - measure of fit (area)

[ ] generation of new claims (**Goodfellow2016?**)

In Figure 10, we perform a grid search across the factor space. For each point in the grid search, we generate a text by passing the point through the decoder and the resulting embedding as an input into the LLM. In the plot, we have three types of different markers. A blue bubble with a number represents the location of the respective claim, while a colored dot represents a point where we generate one of the points that are part of the training data. Red crosses are points from which we generate a string, that comes to an end with the end of text token. Whitespace represents points where we do not generate a string that comes to an end within the number of tokens that we consider.

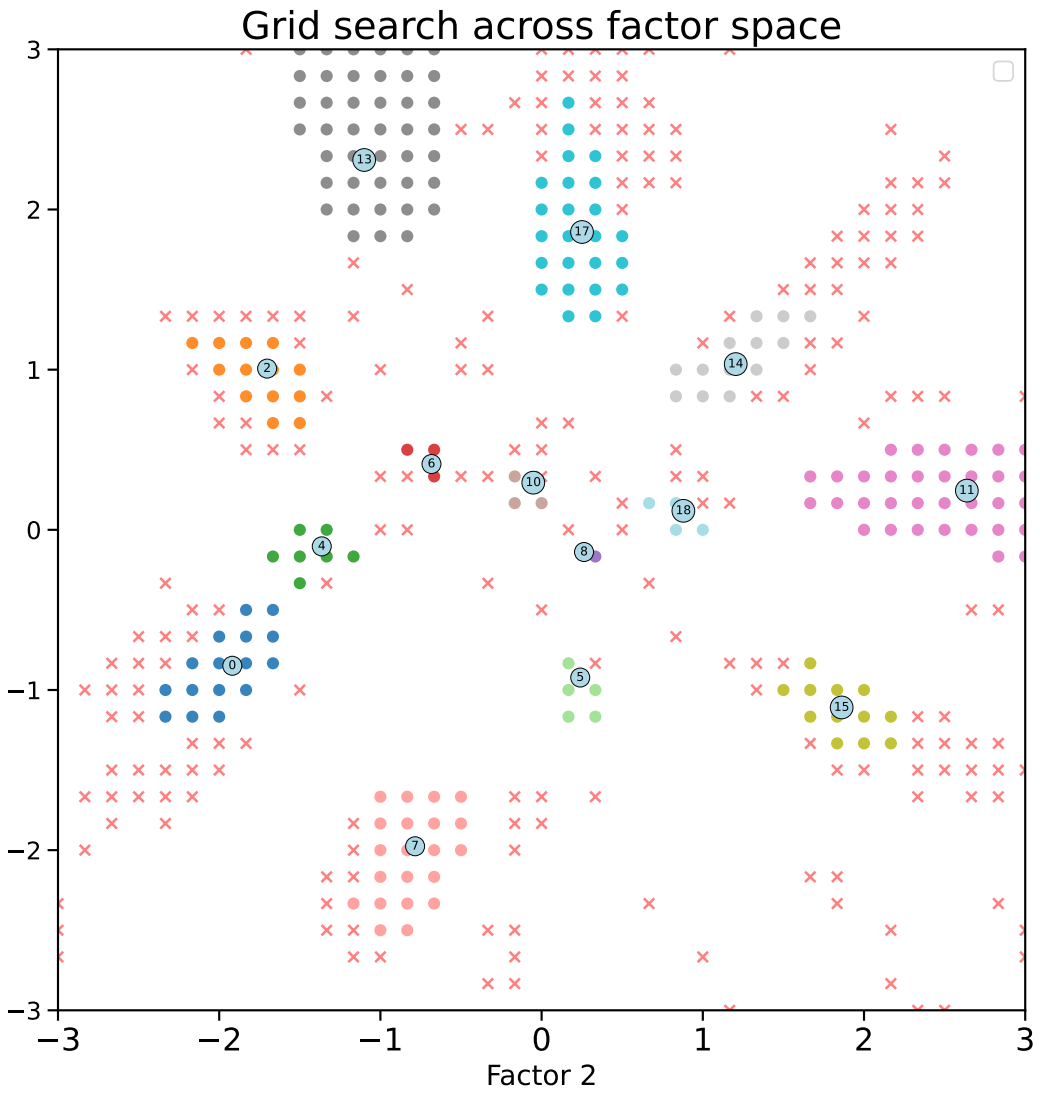


Figure 10: Exploration of Encoding Space

We explore the generation based on our summary embedding deeper, by analysing the generation tree when generating with the summary embedding. We expect that at each generation step, the probability for

the correct token approaches one, while the probability for all other tokens goes to zero. In Figure 11, we visualize the next token probabilities at each generation step for the tokens of the vocabulary. We see that indeed the probability for the correct token is close to one, while the probability for all other tokens is close to zero. In Table 3, we show the generation probabilities for the three most likely tokens at each generation step. This example shows the claim, for which the probability of the correct token in the first position is the lowest out of all claims.

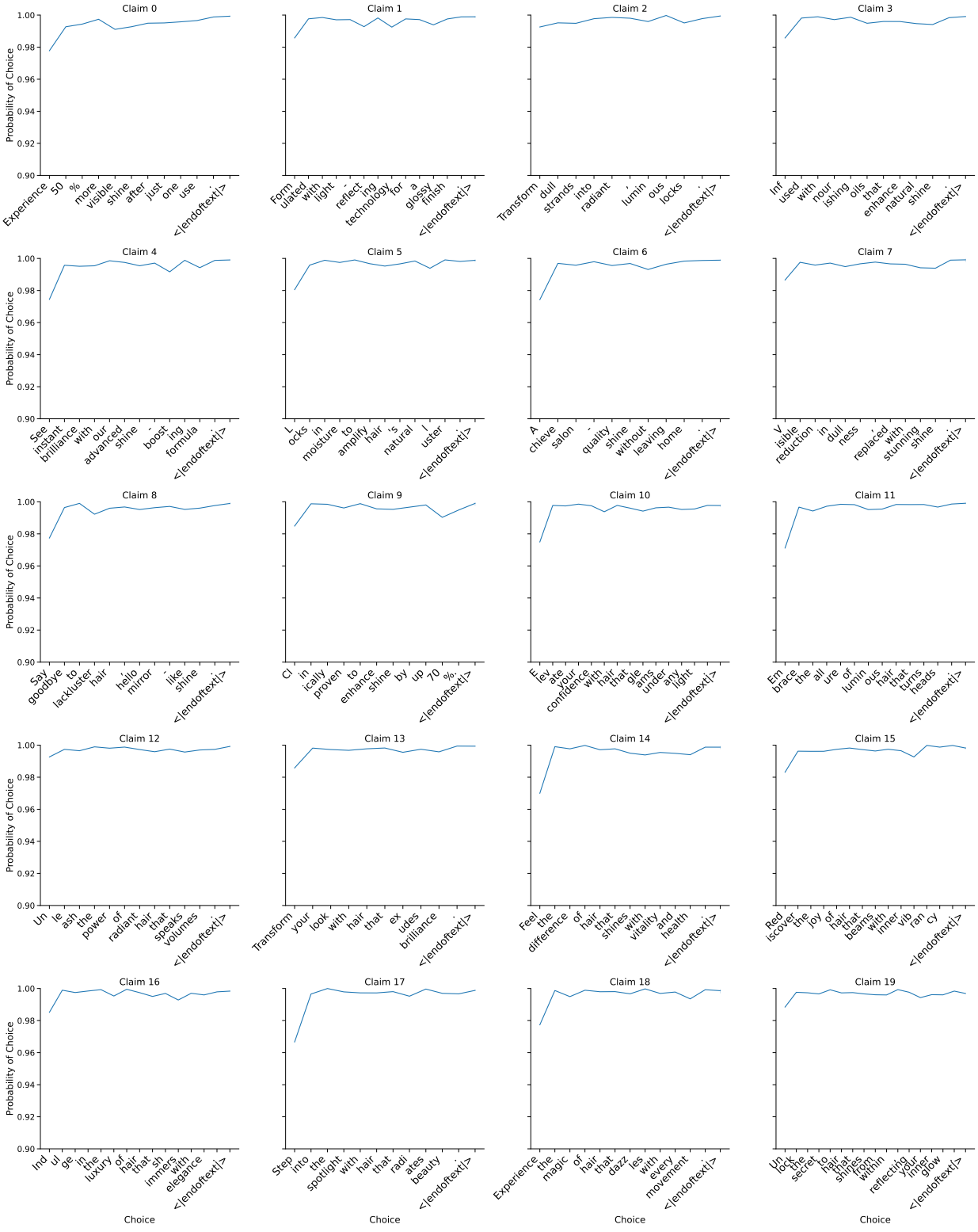


Figure 11: Conditional generation probability for the most likely token at each step. All tokens match the target claims.

Table 3: Example for a low 1st probability

	3rd	2nd	Choice	Prob. 3rd	Prob. 2nd	Prob. Choice
0	Feel	Un	Step	0.0287145	0.0344409	0.887634
1	onto	back	into	0.000927151	0.00125316	0.992544
2	to	class	the	5.98035e-05	6.91189e-05	0.99934
3	’	sun	spotlight	0.000623843	0.00123234	0.992407
4	into	in	with	0.00192485	0.00199383	0.992454
5	a	hairs	hair	0.000946804	0.000969693	0.995565
6	above	.	that	0.000743194	0.0014022	0.993072
7	X	can	radi	0.000538268	0.00169703	0.992411
8	ating	ats	ates	0.000915091	0.00206985	0.996706
9	heat	into	beauty	0.00111988	0.00228065	0.987492
10	).	.)	.	0.000529124	0.00114012	0.996219
11	*2x line-break*	]	*eos-token*	0.000603324	0.00100365	0.995909
12	*2x line-break*	*linebreak*	*eos-token*	7.61097e-06	0.00124311	0.99872
13	*2x line-break*	*linebreak*	*eos-token*	2.16886e-05	0.000565556	0.999231
14	*2x line-break*	*linebreak*	*eos-token*	3.80865e-05	0.00084979	0.998713
15	*2x line-break*	*linebreak*	*eos-token*	5.16162e-05	0.00161664	0.997724
16	*2x line-break*	*linebreak*	*eos-token*	0.000101122	0.0031923	0.995816

Table 4: Example for similar tokens

	3rd	2nd	Choice	Prob. 3rd	Prob. 2nd	Prob. Choice
0	See	Form	Experience	0.00921188	0.013993	0.943786
1	20	25	50	0.00203264	0.00445767	0.976235
2	%.	.	%	0.00142157	0.00205115	0.994173
3	better	More	more	0.000946993	0.00160339	0.996037
4	shine	protective	visible	0.00178126	0.00218073	0.979003
5	glow	light	shine	0.000826363	0.00209097	0.983356
6	when	by	after	0.00171736	0.00367565	0.987
7	a	well	just	0.000570301	0.00172336	0.98959
8	two	a	one	0.00153683	0.00186334	0.989587
9	shine	usage	use	0.000427258	0.00324287	0.990624
10	,	!	.	0.000354906	0.000630647	0.9981
11	.		*eos-token*	0.000454232	0.00156762	0.994366
12	Transfer		*eos-token*	0.000179846	0.00032993	0.998234
13		More	*eos-token*	0.000109437	0.000194042	0.998748
14	More		*eos-token*	7.64376e-05	0.000174342	0.999236
15	Lab		*eos-token*	5.29115e-05	0.000444251	0.998936
16	Lab		*eos-token*	0.000276325	0.00155492	0.997141

Empirical Application

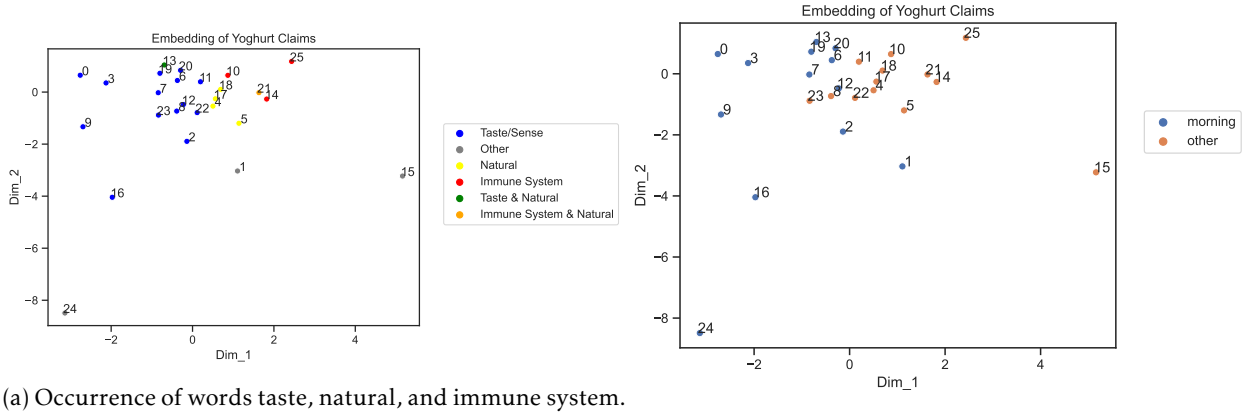
- Summary statistics of the data
- Application to MR data

MR 5

Since we cannot disclose the actual advertising claims, we generate a set of similar claims with ChatGPT. However, we performed all computations on the actual data.

Artificial examples for the yoghurt drink advertising claims:

- *"A burst of fresh flavor to energize your morning"*
- *"Refresh your day with a lively new taste"*
- *"Dynamic flavor for an invigorating start"*
- *"Begin your day with a crisp and revitalizing taste"*
- *"Revitalize your senses with a pure, fresh flavor"*



(a) Occurrence of words taste, natural, and immune system.

(b) Occurrence of words relating to morning theme

Figure 12: Embedding of the yoghurt claims, colored by word features.

- establish that the space is continuous, word features etc.

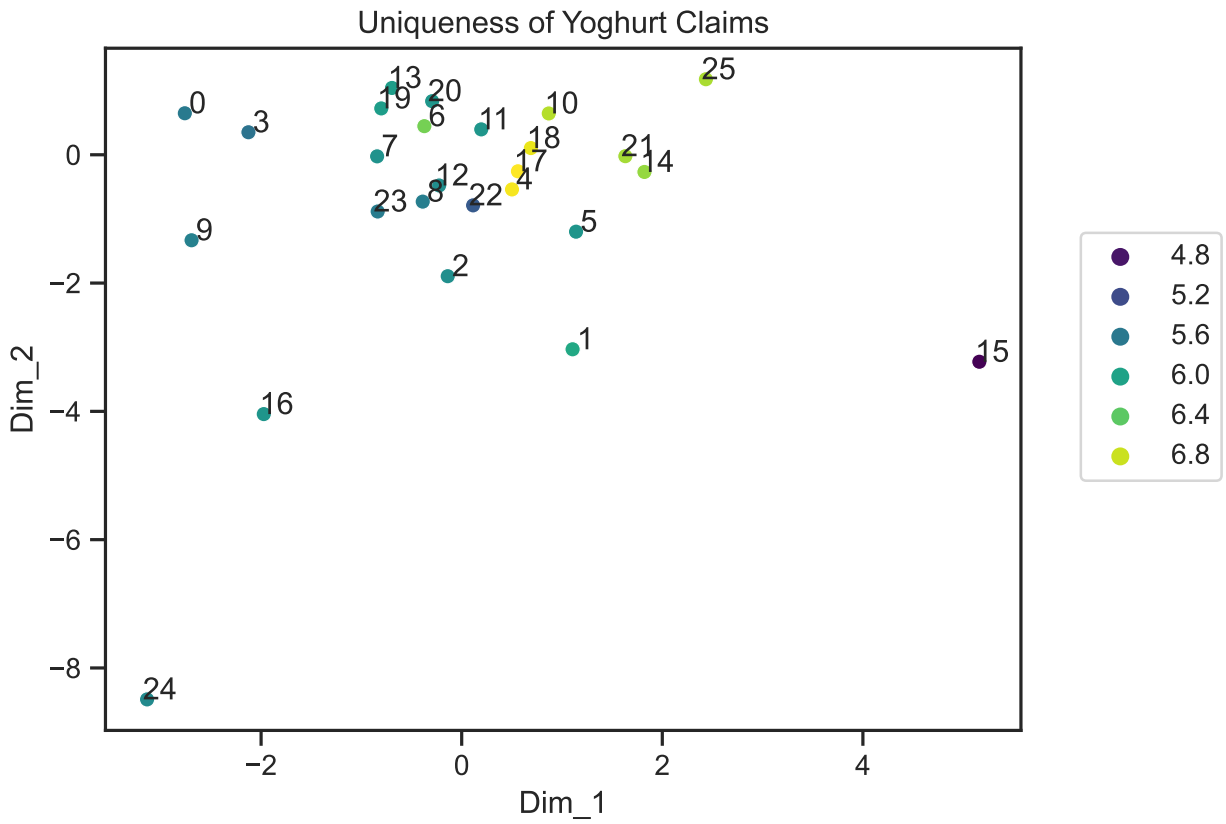


Figure 13: Uniqueness score of embedded yoghurt drink claims.

Figure 12 shows the encoding space for the yoghurt drink advertising claims, and illustrates how the embedding picks up on language features of the documents. Namely, Figure 12a colors the claims based on whether they contain the words "taste" or "sense", "natural", and/or "immune system". We color-coded these three by the basic colors (yellow, red, blue), and claims that contain multiple of these words by the mixtures of these colors. If a claim contains none of these words, we code it in grey. Claims with the same word features cluster together, while claims that contain none of these words, are pushed to the outside of the plot. Intuitively, we would expect that claims which contain multiple of these word features from a transition between the claims which only have one of the word features. For this, we only have claims 13 and 21 as examples, whereas the former does not form such a boundary, and the latter is at the edge of the "Natural" and "Immune System" class.

We expect advertising claims that are more unique to have a more isolated position in the encoding space, as they are less similar to other claims. In other words, the uniqueness score should be positively correlated with the euclidean distance of an encoded claim to its neighest neighbor. Figure 13 shows that this is not the case, the correlation of these two measures is substanial and negative (−0.4375). The opposite is true: The three most unique claims (4, 17, 18) cluster together closely, and the further claims are away from these three, the less unique they tend to be. Claims with low uniqueness scores are spread further apart and far away from the most unique claims. These three most unique claims have some things in common, all of these use a combination of the words "natural", "active", and talk about ingredients (with synonyms). On the other hand, claim 15 is the only claim in the dataset using the word "yoghurt drink" and is also the only claim about emotions. Thereby, it is unique from a language perspective (far away from other points in the encoding), but perhaps not unique to consumers, as there might be similar claims on the market already. Also, uniqueness is highly correlated (0.8843) with the overall rating of claims, some of this correlation could be due to consumers viewing uniqueness as a proxy for "satisfaction". Lastly, the researched advertising claims are self-selected, and perhaps more unique then the average claim that is on

the market already. This might lead to a form of Simpson's paradox (Sprenger and Weinberger 2021), where in the population there is a positive correlation between uniqueness and unique language, but perhaps not in this special sub-sample of the data.

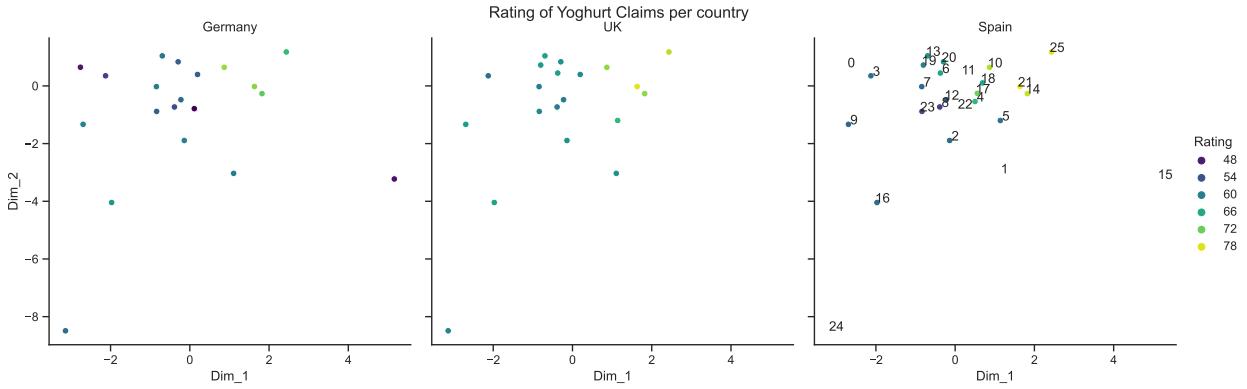


Figure 14: Overall rating of embedded yoghurt drink claims.

Figure 14 again shows the encoding space, but this time colored by the overall rating of the claims. Claims of similar rating tend to cluster together, with higher ratings in the north-east and lower ratings towards the south-west of the graph. The clustering of similarly rated claims, could be due to these claims being similar in language. For marketers, there are two insights from this: One, getting the overall theme of a claim right, can ensure that customers perception of this claim is within a certain ballpark. Two, after identifying a fruitful theme for the advertising claim, it is still useful to explore this neighborhood in more fine-grained steps, as locally, there might be small alterations that have large effects on the perception: See e.g. claims 21 (rating in top 4% percentile) and 14 (rating in top 12%), despite one of the shortest distances in the encoding space. Figure 14 also shows how ratings of uniqueness can differ by country, the reasons for this could lie in what consumers are used to from the domestic market and in cultural differences.

MR 6

Managerial Implications

- Generative AI for personalization
- Brands
- Wording: Ballpark, finetuning idea
- Interactive dashboard to explore the embedding space
- Costs of tailoring advertising claims
  - When we take current prices for the GPT-4o LLM; June 2024 ($5 per 1 Mio tokens input, $15 per 1 Mio tokens output), then training a dataset of 100 advertising claims, for 5,000 epochs will cost $150. Freelancers on platforms such as Upwork.com charge between $20 and $200 per hour for work on tasks such as SEO optimization, copy writing, and creation of advertising claims.
    - Online services for conjoint studies, such as Conjointly.com and Sawtooth charge a few thousand dollars per year to use their services. Leveraging the resulting insights as much as possible, by using generative AI, could hence make market research more cost effective.
- Generator, predictor compatible embeddings; Generative AI to assist and automate design processes in marketing

- Existing approaches
  - Hong and Hoban (2022) using a hierarchical attention network (HAN)
- Burnap, Hauser, and Timoshenko (2023)
  - architecture of encoder, embedding space, predictor, generator
  - images, VAE & GAN; Use semi-supervised learning of unlabelled and labeled images; These attributes can shape the generation
  - augment the design process, rather than automate it
  - Training requires large resources (2 weeks on multiple GPUs); However, use works on regular laptop
  - Don't have pre-trained model available that can generate
  - Performs better than research clinic, at a fraction of the cost, managers want to adapt this model in the organization
  - Differences
    - piggy-back of pre-trained LLM, can just learn the specifics of the domain rather than the language itself
    - Deterministic generation
    - When using bigger model, benefit of emergent capabilities; Brand, Israeli, and Ngwe (2023); Goli and Singh (2024) indicate that LLMs have ermergent capabilities wrt to consumer preferences (NOTE: Is this a fair point to make? We are getting rid of stochastic decoder in a sense)
- The approaches by Ludwig and Mullainathan (2023) and Burnap, Hauser, and Timoshenko (2023) model an encoding space for images, that can be used for generating new images as well as to create features that inform a supervised machine learning task
- These frameworks consist of three componenets: The encoder, which projects datapoints in an encoding space, the generator, which turns points from the encoding space into new datapoints, and the predictor, which predicts a certain label based on a point from the encoding space (compare terminology Burnap, Hauser, and Timoshenko 2023).
- In our context, these three components perform the following tasks: The encoder represents an advertising claim as a vector, the generator generates and advertising claim based on such a vector, and the predictor predicts the marketeers motivation based on vector from the encoding space.
- Rather than inform product design, Ludwig and Mullainathan (2023), use such a framework to generate hypotheses for why a prisoner might be granted bail or not. For this, they predict judges decisions' from bail hearings, based on the mugshot photo of the defendant (predictor) and learn about the distribution of mugshots with a Generative Adversarial Network [CITE Goodfellow GAN]. To generate new hypothesis, they perform gradient descent on the prediction of the predictor with respect to the encoding data point. They search for the alteration to the image, that changes the predicted outcome the most. By making equal steps along this gradient in opposing directions, they obtain to versions of the same image manipulated with respect to this feature. Ludwig and Mullainathan (2023) also find, that comparing these two images often yields people to name the specific change, creating new hypothesis on why an inmate might be granted bail or not.
- We call the input data, x, and its distribution $p(x)$. In our setting, x is an advertising claim.
- We call labels, y and its distribution $p(y)$. In our setting, a label is e.g. the motivation of the marketeer behind an advertising claim, such as whether they want the claim to be "tangible" or "intangible". The "label" can also be a continuous variable, such as a consumer rating, turning classification problems into regression problems.
- We assume, that we can predict y based on the data x. The distribution of the label given the data is the likelihood $p(y|x)$.
- Generative Adversarial Networks (GANs) consist of two components: The generator and the discriminator. The discriminator takes a datapoint as its input and predicts whether this is a genuine or fake datapoint, while the generator creates new datapoints based on noise. The learning objective for the generator is to create samples that pass the scrutiny of the discriminator, while

the objective for the discriminator is, to correctly classify its test cases. There exists an equilibrium for these two objectives, when the generator creates samples that are indistinguishable from real datapoints. In essence, the GAN learns the distribution $p(x)$ of the data.
  - The GAN architecture was designed for continuous data (images), rather than discrete data (text) , and adaptations if this model are necessary (see De Rosa and Papa 2021)
  - There is a potential to benefit of the pre-trained information in LLMs. Rather than GANs, which require more training data as these networks need to learn the properties of an image from the ground up, we can focus directly on the types of texts that are relevant to our domain.
  - The training objective of LLMs is to predict the next token in a sequence of tokens
  - Need to model $p(x)$ and $p(y|x)$
  - For text data, we have options to model $p(y|x)$, e.g. with a supervised machine learning model. There are also potent ways to represent the input data x, sucha as by pooling BERT embeddings of a text (Devlin et al. 2018; Shen et al. 2018).
  - We can also make draws from $p(x)$, however, this only works through prompt engineering (CITE). And we do not obtain an encoding space, that can be shared with a predictor model.
  - We solve this problem by finding an encoding space, that allows for informed generation and provides a representation useful for supervised learning tasks.
  - We find this encoding space through an optimization. Namely, for every sequence in our dataset, we find an embedding vector, such that when we use this vector as an input to our LLM, we maximize the likelihood of generating this sequence. This we call the summary embedding. We perform this optimization for our full dataset simultaneously by using a factor model.
  - By using this factor model, we create a bottleneck, forcing the model to learn helpful features; In other factor models, these features are often interpretable dimensions (Goodfellow, Bengio, and Courville 2016).

Discussion

- Further Applications
  - Framework of generator and predictor
  - optimization of prompts in continuous space
  - automatic answer evaluation
  - explainable AI for generative models

Problems

- Issues with Generation
  - How to evaluate the quality of the generated ad claims Tatsunori Hashimoto et al. (2019)
  - Another explanaition could lie in the decoding strategie. Ari Holtzman et al. (2020) show how text generation based on a maximum likelihood objective leads to text that sounds bland and not human, propsing the use of different decoding strategies to make the generated text more human.
  - Using embedding rather than identity matrix as the input turns this into an autoencoder, rather than a factor model. According to Goodfellow, Bengio, and Courville (2016), generation from factor models leads to a combination of the learned features in these factors, but not to the generation of sensible data points. Existing, autoencoder based, approaches that have succeeded in generation tasks, might be more suitable for this task (e.g. CITE: Variational Autoencoder).
  - Need deeper architecture, the manifold on which advertising claims live is not a linear plane?
  - does it make sense to have continuous representation of discrete thing? Then there would be other latent variable architectures, such as Boltzman machines

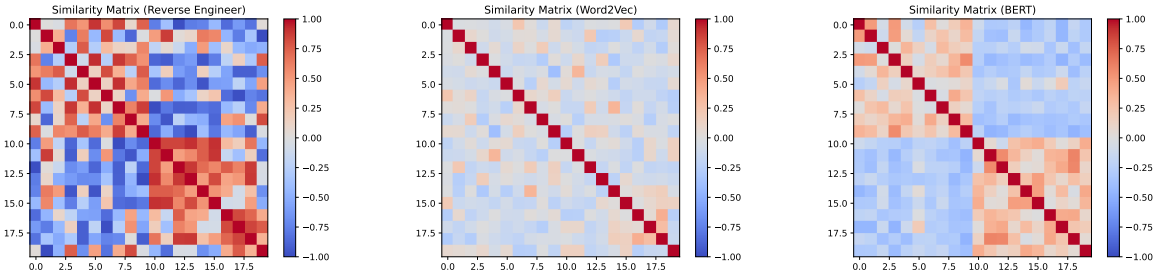
- Implementation challenges
  - Computation time
  - Tokenizer can cause problems?
  - Not as flexible as general LLMs, as we need to have a target sequence and have a use case where we don't want a distribution

Expansion of this research

- Incorporating the context of the product itself, embedding e.g. the brand, product features and competitors in the model, to move into part of the generation space that is "open" for the product
- Using fine-tuned LLMs
- MAB around the fitted AE, to incorporate consumer feedback in online learning: Which areas of the space are valuable for which consumers?
- For online learning, could wrap the decoder into an MAB and let the MAB explore the space by pulling arms
- Could also learn preferences online with this setup
- Better LLM
- Image, video, website, audio data. Perhaps website a good application. Perhaps audio a rather simple way to make this multi-modal (content and sound)

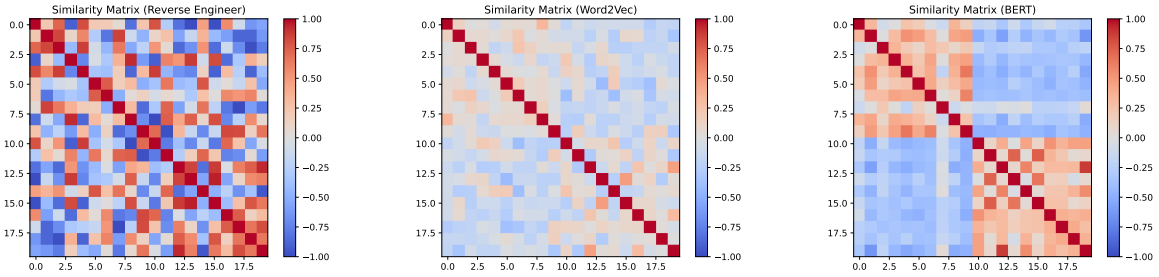
Conclusion

Appendix



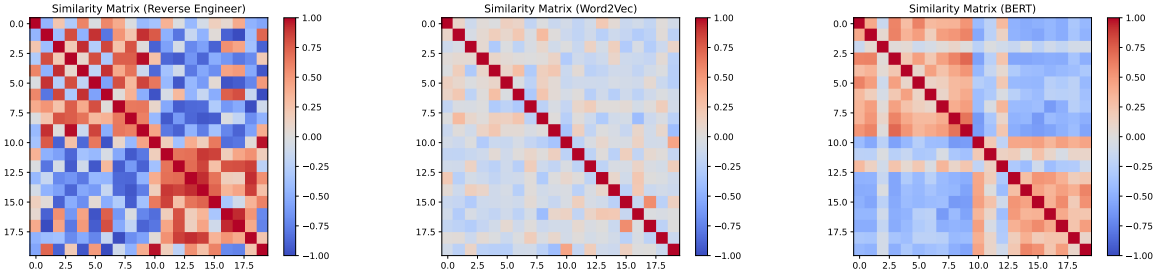
(a) RE (b) W2V (c) BERT

Figure 15: Correlation matrices along the claims.



(a) RE (b) W2V (c) BERT

Figure 16: Correlation matrices along the claims.



(a) RE (b) W2V (c) BERT

Figure 17: Correlation matrices along the claims.

References

Ari Holtzman, Ari Holtzman, Jan Buys, Jan Buys, Leo Du, Leo Du, Maxwell Forbes, Maxwell Forbes, Yejin Choi, and Yejin Choi. 2020. "The Curious Case of Neural Text Degeneration," April.

Brand, James, Ayelet Israeli, and Donald Ngwe. 2023. "Using GPT for Market Research." {{SSRN Scholarly Paper}}. Rochester, NY. https://doi.org/10.2139/ssrn.4395751.

Burnap, Alex, John R Hauser, and Artem Timoshenko. 2023. "Product Aesthetic Design: A Machine Learning Augmentation." *Marketing Science* 42 (6): 1029–56.

Chakraborty, Shayak, and Partha Pakray. 2024. "Abstractive Summarization Evaluation for Prompt Engineering." In *Advances in Visual Informatics*, edited by Halimah Badioze Zaman, Peter Robinson, Alan F. Smeaton, Renato Lima De Oliveira, Bo Nørregaard Jørgensen, Timothy K. Shih, Rabiah Abdul Kadir, Ummul Hanan Mohamad, and Mohammad Nazir Ahmad, 14322:629–40. Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-99-7339-2_50.

De Rosa, Gustavo H, and João P Papa. 2021. "A Survey on Text Generation Using Generative Adversarial Networks." *Pattern Recognition* 119: 108098.

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.

Eggers, Felix, Henrik Sattler, Thorsten Teichert, and Franziska Völckner. 2022. "Choice-Based Conjoint Analysis." In *Handbook of Market Research*, edited by Christian Homburg, Martin Klarmann, and Arnd Vomberg, 781–819. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-57413-4_23.

Goli, Ali, and Amandeep Singh. 2024. "Frontiers: Can Large Language Models Capture Human Preferences?" *Marketing Science*, April. https://doi.org/10.1287/mksc.2023.0306.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. Cambridge, Massachusetts: The MIT Press.

Hong, Jiyeon, and Paul R. Hoban. 2022. "Writing More Compelling Creative Appeals: A Deep Learning-Based Approach." *Marketing Science* 41 (5): 941–65. https://doi.org/10.1287/mksc.2022.1351.

"How to Generate Text: Using Different Decoding Methods for Language Generation with Transformers — Huggingface.co." n.d.

Huang, Zihao, and Tao Chen. n.d. "Enhanced News Summarization Using Large Language Models and Advanced Prompt Engineering."

Jeffrey Pennington, Jeffrey Pennington, Richard Socher, Richard Socher, Christopher Manning, and Christopher D. Manning. 2014. "Glove: Global Vectors for Word Representation," October, 1532–43. https://doi.org/10.3115/v1/d14-1162.

Li, Cheng, Jindong Wang, Yixuan Zhang, Kaijie Zhu, Wenxin Hou, Jianxun Lian, Fang Luo, Qiang Yang, and Xing Xie. 2023. "Large Language Models Understand and Can Be Enhanced by Emotional Stimuli." arXiv. https://arxiv.org/abs/2307.11760.

Liu, Nelson F., Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. "Lost in the Middle: How Language Models Use Long Contexts." arXiv. https://doi.org/10.48550/arXiv.2307.03172.

Lu, Sheng, Irina Bigoulaeva, Rachneet Sachdeva, Harish Tayyar Madabushi, and Iryna Gurevych. 2023. "Are Emergent Abilities in Large Language Models Just In-Context Learning?" arXiv. https://doi.org/10.48550/arXiv.2309.01809.

Ludwig, Jens, and Sendhil Mullainathan. 2023. "Machine Learning as a Tool for Hypothesis Generation." w31017. Cambridge, MA: National Bureau of Economic Research. https://doi.org/10.3386/w31017.

"Meta Llama 3 — Llama.meta.com." n.d.

Onan, Aytuğ. 2023. "Hierarchical Graph-Based Text Classification Framework with Contextual Node Embedding and BERT-based Dynamic Fusion." *Journal of King Saud University - Computer and Information Sciences* 35 (7): 101610. https://doi.org/10.1016/j.jksuci.2023.101610.

Phoenix, J., and M. Taylor. 2024. *Prompt Engineering for Generative AI*. O'Reilly Media.

Radford, Alec, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. "Improving Language Understanding by Generative Pre-Training."

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.

Sprenger, Jan, and Naftali Weinberger. 2021. "Simpson's Paradox."

Tatsunori Hashimoto, Tatsunori B. Hashimoto, Hugh Zhang, Hugh Zhang, Percy Liang, and Percy Liang. 2019. "Unifying Human and Statistical Evaluation for Natural Language Generation." *arXiv: Computation and Language*, April.

Tomáš Mikolov, Tomas Mikolov, Ilya Sutskever, Ilya Sutskever, Kai Chen, Kai Chen, Kai Chen, et al. 2013. "Distributed Representations of Words and Phrases and Their Compositionality" 26 (December): 3111–19.

Tomáš Mikolov, Tomas Mikolov, Kai Chen, Kai Chen, Kai Chen, Greg S. Corrado, Greg S. Corrado, J. Michael Dean, and Jeffrey Dean. 2013. "Efficient Estimation of Word Representations in Vector Space," January.

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.

Wei, Jason, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, et al. n.d. "Emergent Abilities of Large Language Models."

Yinhan Liu, Yinhan Liu, Myle Ott, Myle Ott, Naman Goyal, Naman Goyal, Jingfei Du, et al. 2019. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." *arXiv: Computation and Language*, July.

Zamfirescu-Pereira, JD, Richmond Y Wong, Bjoern Hartmann, and Qian Yang. 2023. "Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts." In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 1–21.