Deconstructing Retrieval Abilities of Language Models

Gottfried Wilhelm Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Data Science
Fachgebiet Wissensbasierte Systeme
Forschungszentrum L3S

Thesis by
Hauke Tristan Hinrichs

First examiner:	Prof. Dr. Wolfgang Nejdl
Second examiner:	Dr. Simon Gottschalk
Advisor:	Jonas Wallat

Hannover, 25.05.2023

Abstract

Large pre-trained language models such as BERT dominate many areas of natural language understanding, including information retrieval (IR). Recently a focus of research has been on using the expressive contextualized representations of bi-encoders to efficiently perform single-stage retrieval, replacing traditional term-based approaches. The question arises as to what information these models leverage to get ahead of traditional approaches performance-wise. Since IR determines which information we get to see, e.g., when searching the web, we stress the urgency of validating whether these models perform the task dependably. The probing paradigm is a versatile mechanic to investigate which information is encoded in the representations. However, it fails to answer the question if the encoded information is used by the model. To overcome this limitation, we employ a *causal probing* approach. First, we validate the feasibility of applying *causal probing* to bi-encoders in the context of IR. Then, we test for the importance of several properties related to IR. Similar to probing, we construct property-specific datasets but use them to learn a projection that removes linear information about the property. The projection is used to produce counterfactual representations. Then, we measure the effect of the intervention on the retrieval performance of our subject model layer-wise. We find that our studied bi-encoder, mainly in later layers, relies on linear encoded information of the following properties: Term importance w.r.t. a query, named-entities, and the type of information need a query conveys.[1]

[1]The code of the experiments is available here: `https://github.com/Heyjuke58/causal_probing`

Contents

1. Introduction

As of today, machines can understand text at a level comparable to humans[1]. Their advantage lies in the ability to process vast amounts of text, for which a human would need a considerable period of time to read. Numerous benchmarks in natural language processing (NLP) have been shifted due to the development of large language models (LLMs), with BERT (Devlin et al., 2019) being the most prominent example. The subfield of information retrieval (IR), which will serve as a case study throughout this thesis, has been no exception to this (Lin et al., 2021a).

Performing well on useful tasks is a start, but there is an urgent need (Adadi and Berrada, 2018) to understand the inner workings of these de facto black boxes, which all deep neural networks (DNNs) essentially are. Considering that LLMs used for IR de facto decide what information we get to see, e.g., when searching the web, we need to build trust in their accountability. The explainability of DNNs in high-stakes scenarios, such as medicine or the judiciary, is even enforced by the EU Artificial Intelligence Act[2].

Therefore, numerous methods to explain the behavior of neural models in the context of IR have been developed over the past few years. Axiomatic analysis tests whether the models adhere to axioms used for traditional term-based IR (Rennings et al., 2019; Câmara and Hauff, 2020; Völske et al., 2021). Another approach is feature attribution. By perturbing input features and measuring the impact on the model output, importance scores can be assigned to input features (Qiao et al., 2019) or a surrogate model can be trained that locally explains the behavior of the underlying model (Singh and Anand, 2019; Verma and Ganguly, 2019). Probing is yet another approach for interpreting IR models that is also popular outside the IR setting to explain the inner workings of LLMs (Adi et al., 2017; Conneau et al., 2018; Tenney et al., 2019b). In probing, a small classifier called *probe* is trained on the fixed representations of a language model to perform an auxiliary task predicting syntactic, semantic, or structural properties of text. The probe's performance is understood to reflect the degree to which the model has encoded information about the auxiliary task. However, more recent research has shown that drawing causal explanations of model behavior based on the performance of a probe might be misleading. First, the encoding of properties can happen accidentally and independently from the task (Ravichander et al., 2021). Second, expressive probes might simply learn the task instead of relying on decoding the information from the

[1] `https://learn.microsoft.com/en-us/archive/blogs/stevengu/microsoft-achieves-human-performance-estimate-on-glue-benchmark`

[2] `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206`

fixed representations (Hewitt and Liang, 2019).

For us humans, *causality* plays a key role in our understanding of the world (Ganguly et al., 2023). Therefore, we address the weaknesses of conventional probing by performing *causal probing*. Our approach resembles the one proposed by Elazar et al. (2021). Similar to probing, we construct auxiliary task datasets for retrieval properties, but instead of training a probe, we train a projection. In a minimax game (Ravfogel et al., 2022a), this projection learns into which subspace it has to project to guard the property against being predicted by a probe. In a white-box manner, we use the obtained projection to perform a counterfactual intervention on the hidden representations of our subject model. Hopefully, this enables us to observe how the behavior of the model changes when it does not have access to a certain property.

In modern text retrieval systems emerged the trend to split up the task into two stages: First-stage retrieval and re-ranking (Chen et al., 2017). While contextualized language models are particularly useful for the re-ranking step, first-stage retrieval has often been done with traditional term-based text retrieval approaches, mainly due to efficiency reasons. However, more recent research moves in the direction of leveraging semantic models for this task (Guo et al., 2022). The bi-encoder is an architecture that enables the pre-computing of document representations addressing the efficiency issue of first-stage retrieval (Lin et al., 2021a). Accordingly, our subject model will be a BERT-based bi-encoder model that is capable of performing even single-step retrieval, sacrificing a portion of the precision stemming from the multi-stage process for efficiency.

1.1. Contribution

We investigate how our subject model, the bi-encoder TCT-ColBERT (Lin et al., 2020), is influenced by the counterfactual removal of several retrieval properties in performing single-stage retrieval. The properties we examine are the following: Exact-term matching (BM25), semantic similarity of queries and documents (SEM), term importance (TI), named-entity recognition (NER), coreference resolution (COREF) and question classification (QC). We use the passage retrieval dataset from the TREC 2019 Deep Learning Track to synthetically construct the auxiliary task datasets and evaluate our subject model on their test set of 43 test queries.

Additionally, we perform preliminary experiments to validate the proper functioning of our approach (causally probing a bi-encoder in the IR setting). Also, we investigate our used removal technique *Linear Adversarial Concept Erasure* (Ravfogel et al., 2022a).

We seek to answer the following research questions:

RQ 1 Can we confirm the feasibility of *causally probing* our bi-encoder subject model in the context of retrieval?

RQ 2 On which properties does our bi-encoder rely upon to solve the task of text retrieval?

RQ 3 At which layers are important properties encoded?

1.2. Thesis Structure

- Chapter 2 provides the reader with a brief overview of information retrieval (IR) and traditional term-based approaches. In addition, we explain the architecture of the transformer model and how the BERT and its variants, especially TCT-ColBERT since it will be our subject model, are built and trained.
- Chapter 3 presents previous work on the topics of explaining neural IR and probing language models.
- Chapter 4 explains our *causal probing* setup, the retrieval properties we investigated, and the external datasets we used.
- Chapter 5 presents feasibility studies we conduct to validate the applicability of *causal probing* and to gain insights about the removal technique we use. Also, we present and discuss their results.
- Chapter 6 presents and discusses the results of our *causal probing* experiments.
- Chapter 7 concludes the main findings of this thesis and provides possible future work.

2. Foundations

It is assumed that the reader is familiar with the basics of machine and deep learning. If not, the books of Kelleher et al. (2015) and Goodfellow et al. (2016) can be consulted for an introduction and overview of these topics.

2.1. Information Retrieval

Information retrieval (IR) is the process of finding material of unstructured nature that satisfies an information need in a large collection, also referred to as a corpus (Manning et al., 2008). The material can have various modalities ranging from text to images, sounds, or videos. In this thesis, we will focus on textual IR where the elements in the corpus are called documents. They are of unstructured nature in the sense that they do not share a common schema like data in a relational database – they are simply arbitrary texts. The information need is expressed via a user-defined textual query. The most prominent example of IR is searching the web, where the corpus changes highly dynamically. For the problem statement, we relax this by assuming a fixed corpus. We will now formalize the task of IR and elaborate on the difficulties web search engines face, beyond the ever changing corpus.

Let Q be the set of all possible queries and $\mathcal{C}$ the corpus. Assume there exists a relevance function $r : Q \times \mathcal{C} \to \mathbb{R}$ that assigns a score to each query-document pair (q, d), satisfying the following: For every $q \in Q$ and every $d_i, d_j \in \mathcal{C}$ it holds that $r(q, d_i) > r(q, d_j)$ if d_i is more relevant than d_j with respect to q. Then, the only thing we would need to do, to solve the task of IR is to calculate all relevance scores for the given query and order the results.

However, there exist two problems concerning this relevance function: First, scoring the relevance of a document regarding a query is not a trivial task due to several caveats. Second, depending on the relevance function and the size of the corpus, calculating the relevance scores of all documents for a single query might be computationally infeasible.

To elaborate on the first problem we use an example. The query "Data science application" most likely refers to data science being applied to solve a problem. However, the user might have meant to ask "How to apply for a data science job?". There can be a discrepancy between the query and the actual information need. This phenomenon can be triggered by the known habit of users to preferably perform keyword searches instead of natural language queries (Azad and Deepak, 2019; Lau and Horvitz, 1999). The so-called *vocabulary problem* (Furnas et al., 1987) occurs since there is a mismatch in the understanding of a word the user prompted and the search engine looks for. The discrepancy ultimately impedes the engine from correctly determining relevance. A way to mitigate this problem is *query expansion* (QE). In fact, the technique is also used to alleviate an array of other problems by enriching the query with context-related words or synonyms either automatically or with user involvement, correcting misspelled words, expanding abbreviations to their full form, or stemming query terms. Azad and Deepak (2019) provides a survey of past and current research on this topic.

A popular approach to mitigate the computational efficiency problem is to break up the retrieval into two stages: First-stage retrieval and a re-ranking step. The task of the former is to efficiently retrieve k possibly relevant documents with good recall, such that the latter can order these k documents by relevance with high precision. Neural ranking models based on pre-trained large language models like BERT have shown an astonishing capability to perform the re-ranking task (Nogueira and Cho, 2019). As they only need to consider the previously retrieved k documents, performing cross-attention for all the k query-document pairs becomes feasible.

A favored choice for the first-stage retrieval has been term-based bag-of-words (BoW) approaches. They ignore word ordering and are capable of representing even large documents compactly. By using an inverted index, they perform very efficiently even on large corpora (Manning et al., 2008). Within the next sections, we will describe a prominent candidate for this kind of retrieval, namely BM25, and its building blocks TF and IDF as well as a related technique called TF-IDF.

2.1.1. TF-IDF

Term frequency-inverse document frequency (TF-IDF) assigns a weight to term t depending on a document d and corpus $\mathcal{C}$ statistics, reflecting its importance. Additionally, to obtain a measure of relevance for a document to a query q, the TF-IDF weights corresponding to the terms in the query are summed up. TF-IDF is composed of two parts:

1. Term frequency (TF): Assuming a document d is more relevant, the more often a query term t occurs in it, one can compute the following value:

$$\mathrm{TF}(t, d) = \frac{f_{t,d}}{|d|} \tag{2.1}$$

where $f_{t,d}$ is the raw count of t in d, and $|d|$ is the total number of terms in d (with repetitions), adjusting for the factor that longer documents might contain terms more often than shorter ones. Still, one can argue that a linear relationship between relevance and term occurrences is unrealistic. Hence, there is also the possibility, among others, to use logarithmic scaling. In this case, term frequency calculates as follows:

$$TF(t,d) = \begin{cases} \log(1 + f_{t,d}) & \text{if} f_{t,d} \neq 0 \\ 0 & \text{otherwise} \end{cases}. \tag{2.2}$$

2. Inverse Document Frequency (IDF): Linguistically common terms like articles, conjunctions, or prepositions, when contained in both a query and a document, are generally not an indicator of relevance. Therefore inverse document frequency introduces a notion of how important a word is based on the ratio of documents it appears in. The less often a term appears relative to the corpus size, the more important it is, and vice-versa. IDF computes as follows:

$$\mathrm{IDF}(t, \mathcal{C}) = \log \frac{|\mathcal{C}|}{df_t} \tag{2.3}$$

where $|\mathcal{C}|$ is the total number of documents in the corpus and df_t is the count of documents in the corpus that contain term t.

Putting the parts together, the TF-IDF weight is obtained by taking the product of TF and IDF:

$$\text{TF-IDF}(t, d, \mathcal{C}) = \mathrm{TF}(t, d) \cdot \mathrm{IDF}(t, \mathcal{C}) \tag{2.4}$$

The relevance score of a document to a given query calculates as follows:

$$\mathrm{score}_{\text{TF-IDF}}(q, d, \mathcal{C}) = \sum_{t \in q} \text{TF-IDF}(t, d, \mathcal{C}) \tag{2.5}$$

2.1.2. BM25

BM25 extends TF-IDF while achieving the same: Assigning weights to terms and providing a relevance score when summing over query terms while relying on exact term matching. It differs from TF-IDF by introducing two hyperparameters and a normalization term for document lengths. The BM25 term weight calculates as follows:

$$\mathrm{BM25}(t, d, \mathcal{C}) = \mathrm{IDF}(t, \mathcal{C}) \cdot \frac{\mathrm{TF}(t, d)}{k_1 \cdot (1 - b + b\frac{|d|}{\mathrm{avg}|d|}) + \mathrm{TF}(t, d)} \tag{2.6}$$

where $|d|$ is the document length and $\mathrm{avg}|d|$ is the average document length over the whole corpus. $k_1 \in \mathbb{R}$ adjusts how fast the term frequency saturates: With a lower k_1 value BM25 saturates quicker, i.e. fewer occurrences of a term are needed, and vice-versa. By default, k_1 is chosen as 1.2. $b \in \mathbb{R}$ weighs the document length normalization term. With $b = 0$ the normalization is absent, and with $b = 1$ there is full normalization. By default, b is chosen as 0.75 (Robertson et al., 1994; Robertson and Zaragoza, 2009).

More often, the use case for BM25 is to assign relevance scores to documents, e.g. using it as a first-stage retrieval model, where it is a popular choice due to its efficiency and conceptual simplicity.

$$\text{score}_{\text{BM25}}(q, d, \mathcal{C}) = \sum_{t \in q} \text{BM25}(t, d, \mathcal{C}) \tag{2.7}$$

2.2. Transformer

The transformer (Vaswani et al., 2017) is a DNN architecture to perform *sequence-to-sequence* tasks, as it has been developed with the intention to improve machine translation. A sequential input $X = (x_1, ..., x_n)$ gets encoded into dense representations $Z = (z_1, ..., z_n)$ of the same length via an encoder before a decoder generates a sequential output $Y = (y_1, ..., y_n)$ from Z. Therefore, it can be applied to any data that can be provided in a sequential manner, such as images (Dosovitskiy et al., 2021), videos (Neimark et al., 2021), audio (Verma and Berger, 2021), time series (Wen et al., 2022), and text (Devlin et al., 2019; Brown et al., 2020).

In general, the transformer works by utilizing a self-attention mechanism. Members of a sequence are encoded to contextualized representations, by attending to all other members in the sequence. Since the encoding for every member can be generated simultaneously, transformers offer a major efficiency improvement to previously dominant recurrent neural networks (RNNs) in NLP like the long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997). Since the model we will study in this thesis is a close descendant of BERT (Devlin et al., 2019), which only uses the encoder part of the transformer architecture, only the encoder will be explained in depth.

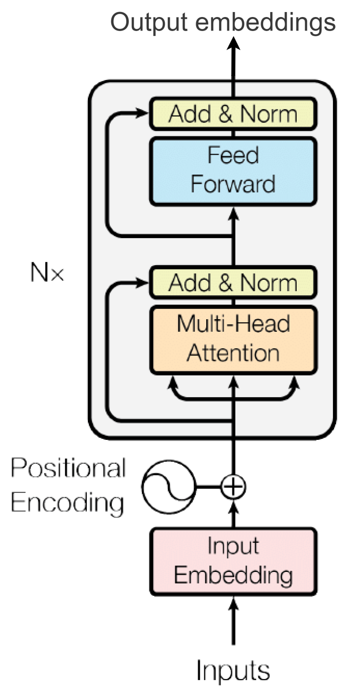

Figure 2.1.: Encoder architecture, taken from Vaswani et al. (2017).

2.2.1. Encoder Architecture

Figure 2.1 depicts the architecture of the encoder by Vaswani et al. (2017). It consists of an input layer in which the input tokens get embedded and a positional encoding is added. Then N blocks of the same structure are stacked onto each other. Each block consists of a multi-head attention layer, a fully-connected feed-forward network, two layer normalizations, and two residual connections. Assuming the input is a list of n tokens, the output of the whole encoder is $Z = (z_1, ..., z_n)$ with $z_i \in \mathbb{R}^d$.

We will now explain each building block in detail starting with the input layer.

2.2.2. Input Layer

First, the input sequence $X = (x_1, ..., x_n)$ is encoded with the help of a learnable token embedding layer. The vocabulary of tokens has to be pre-defined, all tokens that do not correspond to an entry in the vocabulary need to be assigned an *unknown* token.

The embedding layer embeds each token to a d-dimensional dense representation. Then, a positional encoding is added, because the model would otherwise be unable to keep track of the relative and absolute positions of tokens. It has a dimension of d, such that it can be added to the input embeddings. Vaswani et al. (2017) experiment with learned positional encodings but settle for encodings based on sine and cosine functions. Compared to learned positional encodings, they offer the advantage that the model might be able to extrapolate to inputs of longer lengths than seen during training. They calculate them as follows:

$$\begin{aligned} PE_{(pos,2i)} &= \sin(\frac{pos}{1000^{\frac{2i}{d}}}) \\ PE_{(pos,2i+1)} &= \cos(\frac{pos}{1000^{\frac{2i}{d}}}) \end{aligned} \tag{2.8}$$

where pos is the position of the token in the sequence and $i \in \{1, ..., d\}$ is the models hidden dimension.

2.2.3. Multi-Head Attention Layer

This section will explain how the multi-head attention layer works. It uses a mechanism which is called self-attention. We will start with explaining the original idea of attention.

Attention Mechanism

Bahdanau et al. (2015) propose a mechanism to improve the translation ability of an encoder-decoder architecture based on RNNs. The encoder is a bi-directional and the decoder is a uni-directional RNN. The attention mechanism gives the decoder the ability to selectively consider specific representations of tokens from the input sequence as a context when trying to predict the next token. The motivation is to mitigate a well-known problem of RNNs: They struggle to encode long-term dependencies, due to the vanishing gradient problem (Hochreiter and Schmidhuber, 1997). However, for translating, it might be helpful to account for long-term dependencies because semantically identical sentences from two languages might have wildly differing structures.

Let us consider $s_t = f(s_{t-1}, y_{t-1}, c_t)$ to be the hidden state of the decoder f at timestep t, the last predicted token y_{t-1}, and a context vector c_t. c_t is computed in the following manner:

$$c_t = \sum_{j=1}^{n} \alpha_{t,j} h_j \tag{2.9}$$

where h_j is the concatenated (left-to-right and right-to-left) hidden state of the bi-directional encoder for the token at position j, n is the number of tokens in the input sequence, and $\alpha_{t,j}$ is defined in the following way:

$$\alpha_{t,j} = \frac{\exp(a(s_{t-1}, h_j))}{\sum_{k=1}^{n} \exp(a(s_{t-1}, h_k))} \tag{2.10}$$

where a assigns a score to the alignment of the token that is next to be predicted with the j-th input token. It is calculated considering the last hidden state of the decoder s_{t-1} and h_j. a is a feed-forward neural network that is jointly trained together with the encoder-decoder architecture. The work from Luong et al. (2015) propose other options for a, e.g., the dot product. However, in the here discussed setting, the vectors do not have an equal length, so the dot product is not applicable.

Self-Attention

Cheng et al. (2016) adapt this concept to what they call *intra-attention*. Now, not only the decoder should attend to tokens from an input sequence. The encoder itself is given the ability to attend to all of the other tokens when calculating the representation for a token, including that token itself. Vaswani et al. (2017) extensively rely on this mechanism and remove the bottleneck of using sequentially operating RNNs. The parallelization of language translation seems to promise leaps in efficiency and effectiveness. We will now state how Vaswani et al. (2017) calculate self-attention.

The attention function maps a query and a set of key-value pairs to an output vector, where the query, keys, and values are also vectors. The query, as well as the key, in this context, are tokens from the input sequence. One can understand the terms by considering an analogy to IR: The query is mapped to terms (keys) from a document (value). If the query fits a term, the attention function weights the document higher.

Let $X \in \mathbb{R}^{n \times d}$ be the embedded input. Then, X is multiplied with learnable matrices, corresponding to the concepts of queries, keys, and values, $W^{(q)}, W^{(k)} \in \mathbb{R}^{d \times d_k}$ and $W^{(v)} \in \mathbb{R}^{d \times d_v}$, with $d_k, d_v \leq d$.[1]

$$\begin{aligned} Q &:= XW^{(q)} \\ K &:= XW^{(k)} \\ V &:= XW^{(v)} \end{aligned} \tag{2.11}$$

[1] In their work, Vaswani et al. (2017) choose $d = 512$ and $d_k = d_v = 64$.

An attention matrix is then computed as follows:

$$\text{SelfAttention}(X) = \text{Softmax}(\frac{QK^\top}{\sqrt{d_k}})V \tag{2.12}$$

The scaling by $\frac{1}{\sqrt{d_k}}$ prevents saturation of the softmax, which would lead to small gradients possibly slowing down or even preventing training. Compared to the feed-forward network used by Bahdanau et al. (2015), the dot-product approach from Equation 2.12 offers considerable speed and space advantages, since it solely relies on matrix multiplications.

Multi-Head Self-Attention

The self-attention from Equation 2.12 can be extended to multi-head self-attention by parallelizing the process with separate learnable matrices. Therefore, the self-attention for head $i \in \{1, ..., h\}$ is computed by:

$$\text{SelfAttention}(X, i) = \text{Softmax}(\frac{XW_i^{(q)}(XW_i^{(k)})^\top}{\sqrt{d_k}})XW_i^{(v)} \tag{2.13}$$

Then, the self-attention for all heads are concatenated and projected back again to dimensionality d:

$$\text{MultiHeadSelfAttention}(X) = (\|_{i=1}^{h}\text{SelfAttention}(X, i))W^{(O)} \tag{2.14}$$

where $W^{(O)} \in \mathbb{R}^{hd_v \times d}$ is also a learnable matrix and $\|$ is the concatenation operation. Multiple heads have been shown to be beneficial because they allow the transformer to jointly attend to different representation subspaces at different positions.

2.2.4. Feed-Forward Network

The fully-connected feed-forward network operates point-wise, i.e., is applied to every token representation x separately. It consists of the following components:

$$\text{FFN}(x) = \text{ReLU}(xW^{(0)} + b^{(0)})W^{(1)} + b^{(1)} \tag{2.15}$$

where $W^{(0)} \in \mathbb{R}^{d \times d_{\text{hidden}}}$, $b \in \mathbb{R}^{d_{\text{hidden}}}$, $W^{(1)} \in \mathbb{R}^{d_{\text{hidden}} \times d}$ and $b \in \mathbb{R}^{d}$ are learnable parameters.

2.2.5. Residual Connection

Originally, the idea of residual connections was proposed for deep convolutional networks for image recognition. When increasing the depth of an architecture, one can face the *degradation* problem (He et al., 2016). Excessively increasing the depth causes the accuracy of the model to rapidly decline. Therefore, He et al. (2016) introduce residual connections, which add the identity of a previous representation to a later layer in the network, effectively skipping the layers in between:

$$\text{Residual}(x) = \mathcal{F}(x) + x \tag{2.16}$$

where $\mathcal{F}$ represents a layer in the network.

They provide empirical evidence that the *degradation* is not caused by the vanishing gradient problem, as one would initially expect. Rather, they argue that the problem stems from the circumstance that deeper layers often provide only small changes to the representations. Hence, they need to learn a transformation that is almost the identity. They hypothesize that learning to add a small change is easier when a residual connection is introduced, as the layer does not need to learn the identity mapping.

2.2.6. Layer Normilization

By normalizing the activations of neurons to have a zero mean and a standard deviation of 1, the learning process of a DNN can be stabilized and sped up. First, the mean μ_z and the standard deviation σ_z of the current representation $z \in \mathbb{R}^d$ is calculated:

$$\mu_z = \frac{1}{d}\sum_{i=1}^{d} z_i \tag{2.17}$$

$$\sigma_z = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(z_i - \mu_z)^2} \tag{2.18}$$

where z_i is the i-th entry of z. Then, z is normalized by the following formula:

$$\text{LayerNorm}(z) = g \odot \frac{z - \mu_z}{\sigma_z} + b \tag{2.19}$$

where $g \in \mathbb{R}^d$ and $b \in \mathbb{R}^d$ are called gain and bias. They are learnable and can adjust scale and shift of the normalized distribution as needed. $\odot$ indicates an element-wise multiplication.

2.2.7. Transformer Encoder Block

Putting all the previous parts together, the block from Figure 2.1 can be formalized as follows:

$$\begin{aligned} Z &= \text{LayerNorm}(\text{MultiHeadSelfAttention}(X) + X) \\ \text{TransformerEncoderBlock}(X) &= \text{LayerNorm}(\text{FFN}(Z) + Z) \end{aligned} \tag{2.20}$$

where X is either the embedded input with an added positional encoding from the input layer or the output from a previous encoder block.

2.3. BERT and Variants

Now that we established the basic architectural component of BERT, the transformer encoder, we will continue with details on the training regime and the core advantages of BERT (Devlin et al., 2019). After, we will present variants of BERT specialized for the task of retrieval.

2.3.1. BERT

One core advantage of BERT, which it shares with other LLMs (Brown et al., 2020), is the possibility to use its pre-trained version to address a downstream language task like sentiment analysis. Training for a downstream task, also called fine-tuning, requires a smaller dataset compared to the vast text collections that are used to pre-train them. Also, as the pre-trained BERT has already learned syntactic and semantic concepts of language, fine-tuning is considerably quicker than training the model from scratch.

BERT is pre-trained by following a self-supervised training regime consisting of two tasks: Masked language modeling (MLM) and next sentence prediction (NSP). In MLM, a proportion of tokens is replaced with a mask token. The goal of the model is to correctly predict the original token. Compared to objectives for left-to-right language models (Peters et al., 2018a), the MLM approach opens up the possibility to consider the context left and right from the token of interest. The bidirectionality offers an advantage for correctly predicting the token. In the second training objective, namely NSP, BERT has to decide whether two sentences are in order or randomly sampled. The motivation is that BERT learns about relationships of sentences which is important for downstream tasks like question answering.

2.3.2. ColBERT

ColBERT (Khattab and Zaharia, 2020) is a BERT-based model engineered for the task of single-stage text retrieval. As it is the predecessor of the model we will investigate in this thesis (TCT-ColBERT), we first want to present how it operates and share its core ideas. As visible in Figure 2.2 (right-hand side), query and document are encoded separately, making ColBERT a bi-encoder. Bi-encoders can preprocess all documents of a collection query-independent, which can be beneficial efficiency-wise. Contrarily, if BERT is used to determine relevance (see Figure 2.2 left-hand side), it can cross-attend to all query and document terms, removing the ability to preprocess the documents. ColBERT realizes the retrieval by performing a late interaction between the representations of a query and a document. We will start by stating how ColBERT produces the representations for queries and documents. They use the same BERT encoder for both queries and documents. However, for queries, they prepend a signaling token $[Q]$. Also, they pad the query with mask tokens to a fixed length to allow automatic query expansion. For

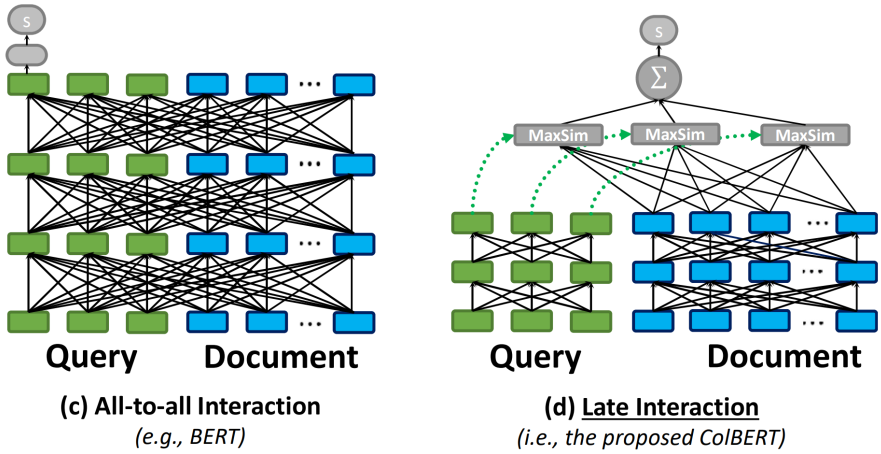

Figure 2.2.: Idea of late interaction to enable bi-encoder structure for IR (right-hand side), as opposed to a cross-encoder (left-hand side). Taken from Khattab and Zaharia (2020).

documents, they prepend a signaling token $[D]$ and do not pad with mask tokens. Then, two separate 1-dimensional convolutional layers (for queries and documents) without activation reduce the dimensionality to a much smaller dimension m than BERT's hidden dimension. After that, the layer is normalized such that every representation has an L2 norm equal to 1. As a last step, the document encoder filters representations of punctuation symbols, which are considered irrelevant for retrieval, to reduce the size of the output further. In summary, the encoders can be described as follows:

$$z_q = \text{Normalize}(\text{Conv1D}(\text{BERT}(\text{"}[Q]q_1...q_l[\text{MASK}]...[\text{MASK}]\text{"}))) \tag{2.21}$$

$$z_d = \text{Filter}(\text{Normalize}(\text{Conv1D}(\text{BERT}(\text{"}[D]d_1...d_m\text{"})))) \tag{2.22}$$

The relevance score between a query q and a document d is computed by the following formula:

$$\text{MaxSim}(q, d) = \sum_{i=1}^{n} \max_{j \in \{1,...,m\}} z_{q,i} z_{d,j}^{\top} \tag{2.23}$$

where n is the number of tokens in the padded query and $z_{q,i}$ is the i-th token representation of the query. Analogously, m is the number of document tokens and $z_{d,j}$ the j-th token representations of the document. The training is carried out with triplets of a query, a relevant document, and a non-relevant document.

2.3.3. TCT-ColBERT

Lin et al. (2020) try to address two major limitations of ColBERT: First, they replace the *MaxSim* operator to calcualte relevance scores from Equation 2.23 with a dot-product.

The calculation of *MaxSim* introduces an overhead that makes single-stage retrieval infeasible at a large scale. Second, they address the issue that ColBERT needs to store multiple representations per document and thus has high storage requirements. Khattab and Zaharia (2020) already try to mitigate this problem by reducing dimensionality and filtering unimportant token representations. Lin et al. (2020) propose to represent each document with a single average pooled vector over all document token representations. To compute the dot-product, queries also need to be represented as a single vector. The training process is carried out via knowledge distillation (Hinton et al., 2015) with ColBERT in the role of the teacher model. Lin et al. (2020) propose a tight coupling between the teacher and the student (TCT-ColBERT), which means that the teacher infers relevance scores during the distillation process, as opposed to pre-computing them. Therefore, the expressiveness of the *MaxSim* operation of ColBERT is distilled into the representations of the student model. Similar to ColBERT, TCT-ColBERT uses a shared BERT encoder for queries and documents and prepends a signaling token ($[Q]$ or $[D]$). In contrast, there are no 1-dimensional convolutions or filtering layers. The token representations are average pooled over the token dimension to obtain a single d-dimensional vector per query or document. In a follow-up paper, Lin et al. (2021b) improve the training with an efficient way to add in-batch negative samples.

TCT-ColBERT in the version proposed by Lin et al. (2021b) will be the studied subject model $\mathcal{M}$ throughout this thesis.

2.4. Probing

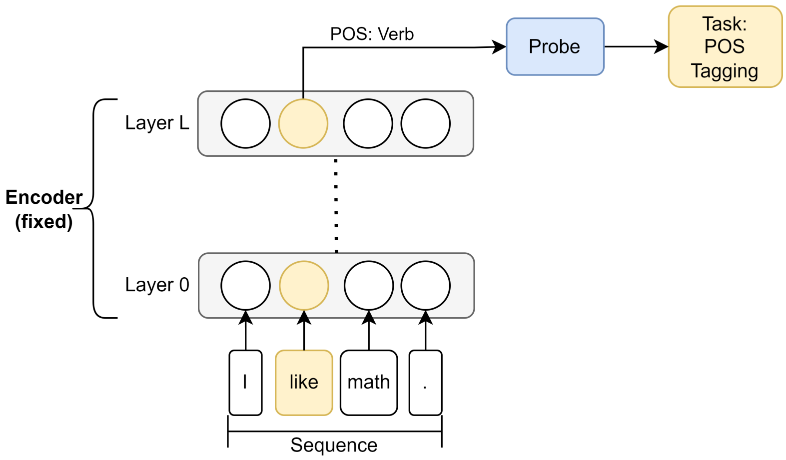

Figure 2.3.: Schematic description of *edge-probing* proposed by Tenney et al. (2019b). Example: Part-of-speech tagging.

Probing is an interpretability technique that measures which information can be decoded from the representations of neural network model. The intuition is that if certain information is encoded the model might have discovered the benefit of relying on the information to solve its task. Therefore, one might gain insights into the inner workings of the model. An example of how the proposed *edge-probing* framework by Tenney et al. (2019b) can be applied is depicted in Figure 2.3. First, a labeled dataset for a task must be constructed, e.g., for part-of-speech tagging, where the goal is to classify words by their type (noun, verb, adjective, adverb, etc.). Then, the task is trained on the fixed representations of the investigated model. Given the specific token representations (yellow circle in Figure 2.3), a small classifier (probe) should predict the part-of-speech label.

3. Related Work

3.1. Probing Language Models

Recent research addressing the mitigation of the opaqueness of neural representations of pre-trained language models covers a broad range of linguistic properties under investigation, spanning syntax, semantics, and structure. A common approach is to apply the probing paradigm. In probing, a simple model, also called a diagnostic classifier, is trained on the representations of a language model to predict a linguistic property. If such a simple model can make an accurate prediction above the majority accuracy, it indicates the representations encode information about the property.

Adi et al. (2017) analyze how well a traditional sentence encoder using the average of word2vec word embeddings (Mikolov et al., 2013) compared to a context-dependent sentence encoder based on recurrent neural networks (LSTM) captures isolated aspects of sentence structure, namely sentence length, word content, and word order. They find that the LSTM encoder is very effective at encoding word order and word content. However, the traditional approach is also surprisingly effective, especially at encoding sentence length and word content.

Similarly, Conneau et al. (2018) create a benchmark consisting of 10 probing tasks, which is supposed to shed light on the linguistic information that pre-trained sentence-encoders encode into their representations. The tasks range from simple structural ones, such as predicting sentence length, to more challenging ones, such as predicting the syntactic tree depth of sentences, to complex semantic tasks, like predicting whether sentences containing two subordinate clauses have been permuted or not. They compare the results to a similar baseline as Adi et al. (2017) and observe results in line with prior findings: While the baseline model performs well on the task of counting words in a sentence, deep contextual models mostly forget about this simplistic property. In turn, they outperform the baseline on more complicated semantic tasks.

Tenney et al. (2019b) propose the *edge-probing* framework. It is characterized by training a probe only on pooled token representations of a sentence described by one or two spans and an assigned label. The spans are sub-parts of the sentences. An example task is named-entity recognition (NER) in which the position of tokens that make up an entity are described by one span. Other tasks like coreference resolution need two spans since it solves the task of deciding whether two mentions refer to the same entity.

Other work expands the analysis of encoded linguistic properties to a layer-wise approach. The motivation is to uncover if there is a hierarchy in which linguistic tasks can be performed by contextual encoders throughout the layers. Peters et al. (2018b) find that lower layers of bi-directional models capture local syntax, such that upper layers can model higher-level semantics such as coreference. With an equal intent, Tenney et al. (2019a) *edge-probe* BERT to test whether its language understanding resembles the classical NLP hierarchy. By comparing the performance on a task with that from the previous layer, they calculate the expected layer at which an example of a property can be correctly labeled by the model. They find that when aggregating over examples, the order of expected layers resembles the traditional NLP hierarchy: Syntactic tasks have a lower expected layer than more complex semantic tasks.

In an analysis building upon the work of Tenney et al. (2019a), Slobodkin et al. (2021) show that there might be factors not considered by Tenney et al. (2019a) that influence the order of what BERT processes first. By binning examples based on their context lengths of spans, they find that for most tasks, the context length is a considerable mediating factor between the task and the expected layer. Since it can lead to misleading conclusions about model behavior, they advocate for the consideration of mediating factors.

However, there are two more factors researchers stress one needs to keep in mind when drawing conclusions based on the accuracy of probes on auxiliary tasks: First, if the probe's complexity is high, it could simply learn the task at hand, instead of decoding the needed information from the representations. Therefore, a probe's selectivity should be a consideration when designing them. Hewitt and Liang (2019) achieve to measure the selectivity by introducing control tasks. Related to this issue, Voita and Titov (2020) propose a new metric that considers a probe's effort to achieve a certain performance. If the effort is high, the probe is considered to be less selective. Second, language models can encode information in their representations that are irrelevant to the task they are trained on, preventing the derivation of causal explanations for model behavior just by having a probe accurately predict properties from the model's representations. This effect is even more prominent when models are initialized with pre-trained word embeddings or fine-tuned for a specific task because this means that the model has been exposed to different objectives, broadening the reasons why a particular property might be encoded in the representations (Ravichander et al., 2021).

Despite the efforts of controlling a probe's selectivity, the major limitation of conventional probing remains: The difficulty of drawing causal explanations of model behavior. Therefore, we pursue an approach similar to Elazar et al. (2021). Their proposed method, *Amnesic Probing*, focuses on whether decodable information is used by the model. They measure the utility of a linguistic property by removing it from the representations and evaluating if the intervention harms the model's ability to solve a task. *Amnesic Probing* is to be understood as an extension to conventional probing since a property is only removable if it is encoded. The property removal procedure they used, *Iterative Null-space Projection* (INLP) (Ravfogel et al., 2020), allows removing linearly present information from neural representations. It is achieved by repeating the following steps: Training a linear probe on the task; calculating the projection onto the null-space of the probe; projecting the feature space onto the calculated null-space; and starting again from the beginning. The procedure stops when the linear probe is unable to predict the property. The final projection is obtained by aggregating all null-space projections. INLP is related to the procedure we will use throughout this work, *Linear Adversarial Concept Erasure* (LACE) (Ravfogel et al., 2022a). Conceptually the methods achieve the same, but the authors claim to have improved the selectivity of removed information, compared to INLP, by reframing the iterative projection onto null-spaces into a minimax game. We will describe the procedure in depth in Section 4.1.

Lasri et al. (2022) used a similar approach as Elazar et al. (2021) to understand how BERT stores and uses information about grammatical number. They construct a setting that ensures that BERT needs to use the information. In particular, it should solve the number agreement task, i.e., choosing the right version of a verb regarding the grammatical number its subject has. By intervening on the hidden representations and removing the concept of grammatical number they can identify the following: BERT relies on linear encodings of grammatical number, and the encodings for verbs and nouns are orthogonal to one another. The transfer of number information from the noun to the verb happens up to layer 8, but with their analysis, they can not determine at which layers exactly.

Another work that investigates the approach of Elazar et al. (2021) is that of Rozanova et al. (2023). As a case study, they investigate BERT-based models solving natural language inference (NLI) tasks. Regarding the method of Elazar et al. (2021), they discovered unexpected behavior: The performance of a controlled downstream task is not influenced by the amnesic intervention, although they control the downstream task to solely depend on the removed features. Consequently, they urge caution to base the claim that a concept is not used by a model solely on the non-existent influence of its removal. In addition, they suggest a variation in which all dimensions are removed *besides* those detected via amnesic probing. They call this variation *mnestic* probing and find that it provides more reasonable results than amnesic probing in their case.

In a long paper, Feder et al. (2021) highlight the importance of counterfactual approaches to causally explain language model behavior, as existing methods only compute shallow correlations, which might lead to misleading interpretations. Similarly to Elazar et al. (2021) they approach the problem by constructing counterfactual textual representations that have forgotten a concept under investigation. In addition, they produce counterfactuals by perturbing the textual inputs of a model.

3.2. Interpreting Neural IR Models

As this thesis focuses on IR as a case study, we will mention works that aim to explain the workings of neural models in this setting.

Rennings et al. (2019) use diagnostic datasets that each represent a retrieval axiom. Historically these axioms helped to improve traditional IR approaches. The datasets enable a model-agnostic approach to test whether neural IR systems learned to follow the axioms. The gained insights can be used to develop better training strategies or architectural designs.

However, Câmara and Hauff (2020) found by following the approach of Rennings et al. (2019) that BERT does not adhere to retrieval heuristics conveyed by the axioms. Still, it outperforms a traditional baseline by a large margin. They conclude that traditional axioms are not helpful in understanding or further improving BERT-based retrieval models.

MacAvaney et al. (2022) contributed by developing a framework to investigate the behavior of neural IR models and using it to conduct an extensive study. The core idea is to follow a pair probing strategy: Pairs of documents that differ primarily in an investigated property are ranked w.r.t. a query to identify model preferences regarding the property. The framework generalizes the approach of using diagnostic datasets (Rennings et al., 2019) by defining 3 categories of tests. Their results show that term frequency of query terms is still important for neural ranking models like BERT. However, not as important as it is for BM25. Also, as they expected, shuffling the word order hurts the performance of neural IR models.

Formal et al. (2021) investigated ColBERT in a re-ranking setting and found that it relies on exact term matching for important words, i.e., words with a high IDF, which is even amplified after fine-tuning. While ColBERT does not satisfy the traditional IR axiom that terms occurring in many documents are less important (IDF effect), it still captures a possibly more nuanced notion of term importance.

In a follow-up work, Formal et al. (2022) investigate the lexical matching abilities of neural IR models for first-stage retrieval on out-of-domain datasets. They discovered that the models fail to correctly generalize in the zero-shot setting regarding the importance of terms.

Wallat et al. (2023) conventionally probe a BERT re-ranking model to test how ranking properties are distributed along its layers. They test for similar properties that we study in this thesis, namely exact term matching (BM25), semantic similarity, or named-entity detection. Their result is that ranking properties are best captured at intermediate layers.

This thesis provides a first attempt to *causally probe* a neural model in the context of IR.

Since we mentioned only a few works, we refer the reader to Anand et al. (2022) for a comprehensive overview of the topic.

4. Approach

The main objective of this work is to investigate whether certain properties that are assumed to be important for retrieval are used by our subject model $\mathcal{M}$. Moreover, we aim to get an understanding at which layers these properties are encoded in the hidden representations.

To achieve this, we employ a counterfactual approach: We study how the prediction of $\mathcal{M}$ differs when it does not have access to a particular property. One option is to perturb the textual inputs, such that they become independent of a property (MacAvaney et al., 2022; Feder et al., 2021). Since this might be difficult or even impossible, depending on the property one is interested in, we resort to a white-box approach, directly interfering with the hidden representations of our subject model $\mathcal{M}$.

For this purpose, we use *Linear Adversarial Concept Erasure* (LACE) proposed by Ravfogel et al. (2022a).

4.1. Linear Adversarial Concept Erasure

The objective of Ravfogel et al. (2022a) was to mitigate potential biases in pre-trained representations by selectively erasing linearly present information about a protected property (e.g., gender) with minimal impact on unrelated information in the representations. To achieve this, they formulate the problem as a minimax game between two linear opponents. One tries to predict the value of the protected property, while the other tries to hinder the prediction by projecting the feature space to a subspace, in which the predictor faces a tougher challenge. At the end of this process, they obtain a projection that is supposed to transform the feature space such that linearly present information about the property is removed. In case the value of the property is a real number, in other words, we are dealing with a regression problem they derived a closed-form solution. For the other case, i.e., classification, they developed a convex relaxation of the problem (R-LACE). We will now formalize both approaches.

4.1.1. Linear Regression (LACE)

Let $D_{\text{LACE}} = \{(x_i, y_i)\}_{i=1}^{N}$ be a dataset consisting of a feature matrix $X = [x_1^\top, ..., x_N^\top]^\top \in \mathbb{R}^{N\times d}$ and target values $y = [y_1, ..., y_N]^\top \in \mathbb{R}^N$. A linear predictor model $f_i = \theta^\top x_i + b$ with parameters $\theta \in \mathbb{R}^d$ and a bias value $b \in \mathbb{R}$ tries to predict y_i. For the sake of simplicity, we omit the bias value b in future formulas, as it can be subsumed into the parameters vector $\theta = (b, \theta_1, ..., \theta_d)$ and prepending a 1 to the feature vectors $x_i = (1, x_{i,1}, ..., x_{i,d})$. The minimax game between the predictor, who minimizes the loss, and its opponent, whose goal is to maximize the loss, can be described as follows:

$$\min_{\theta \in \mathbb{R}^d} \max_{P \in \mathcal{P}_k} \sum_{i=1}^{N} ||y_i - \theta^\top P x_i||^2 = \min_{\theta \in \mathbb{R}^d} \max_{P \in \mathcal{P}_k} ||y - XP\theta||^2 \tag{4.1}$$

where $\mathcal{P}_k$ is the set of orthogonal $d \times d$ projection matrices neutralizing a subspace of rank k. When using an orthogonal projection, points from the original space are projected to their closest counterparts in the reduced subspace, ensuring minimized unwanted damage to the representations.

k is a hyperparameter and needs to be chosen depending on the task. In the experiments of Ravfogel et al. (2022a), the choice of $k = 1$ suffices to eliminate binary gender information from the hidden representations of a BERT model fine-tuned on a profession classification task. We conduct an experiment to test different choices for k in our scenario (see Section 5.2).

The equilibrium point of 4.1 is reached when:[1]

$$P = I - \frac{X^\top y y^\top X}{y^\top X X^\top y} \tag{4.2}$$

Linear regression tries to explain the relationship, i.e., the covariance, between the columns of our feature data X and the target values y. Since the y is one dimensional, the covariance is a single vector and therefore corresponds to one direction. P is built to project onto the nulls-pace of the covariance direction. Thus, applying P to the input removes all information with respect to y. The best estimate that a predictor wishing to predict y from the altered input can make is to model the variance of y.

4.1.2. (Multinomial) Logistic Regression (R-LACE)

In contrast to linear regression, one target value y_i in binary logistic regression is either 0 or 1. In this scenario, the minimax game can be described as follows:

$$\min_{\theta \in \mathbb{R}^d} \max_{P \in \mathcal{P}_k} - \sum_{i=1}^{N} y_i \log \sigma(\theta^\top P x_i) + (1 - y_i) \log(1 - \sigma(\theta^\top P x_i)) \tag{4.3}$$

with σ being the sigmoid function $\sigma(\theta^\top P x_i) = \frac{1}{1+\exp(-(\theta^\top P x_i))}$.

In multinomial logistic regression with C classes, the target values y_i are now represented as one-hot vectors $y_i \in \{0, 1\}^C$. $y_{i,c} = 1$ means the ground truth class for y_i is

[1] The expression in the paper does not exactly resemble equation 4.2 because the second transposition symbol in the numerator is missing, which we think is a typo.

c. The predictor parameters θ must adapt to this by being of the form $\theta \in \mathbb{R}^{d \times C}$, and the bias becomes a vector $b \in \mathbb{R}^C$, which we again omit. The following describes the minimax game in this scenario:

$$\min_{\theta \in \mathbb{R}^d} \max_{P \in \mathcal{P}_k} - \sum_{i=1}^{N} \sum_{k=1}^{K} y_{i,k} \log \frac{\exp(\theta^{(k)\top} P x_i)}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} P x_i)} \tag{4.4}$$

with $\theta^{(i)}$ being the i-th column of θ. $\mathcal{P}_k$ introduces non-convexity into the game, which Ravfogel et al. (2022a) relax by replacing it with its convex hull, called the *Fantope* $\mathcal{F}_k = \text{conv}(\mathcal{P}_k)$ (Boyd and Vandenberghe, 2014). This allows for better convergence of the minimax game, as it turns it into a convex-concave game benefiting from the existence of a unique equilibrium point (Tuy, 2004; Pang and Razaviyayn, 2016). Still, they experienced rotational behavior when performing alternate optimization of the predictor and the opponent. However, when run sufficiently long, eventually a projection P converges, making the predictor perform no better than the majority-accuracy.

4.2. Probing Bi-Encoders

To the best of our knowledge, there is no research yet that applies the probing paradigm to bi-encoders. Therefore, we share our thoughts on how probing bi-encoders compares to probing cross-encoders in our specific scenario. As an example, we look at the *edge-probing* framework proposed by Tenney et al. (2019b). It operates on a text, one or two spans, which indicate the positions of tokens in the text, and a corresponding label. They utilize an attention pooling mechanism (Lee et al., 2017) over the representation of tokens from the spans to ensure a fixed-length input for training a subsequent MLP probe. To probe bi-encoders for linguistic properties, we could follow this exact framework. Ultimately, our bi-encoder consists of two identical BERT-based encoders, of which we could select only one to examine. However, when probing for properties related to retrieval, it is likely that they depend on both the query and the document. Since bi-encoders generate separate contextual representations, and in particular, our subject model $\mathcal{M}$ is trained to compare the average pooled representations of queries and documents (see Section 2.3.3), we argue that the attention pooling mechanism is not the obvious choice in our case. Therefore, we resort to average pooling for consistency even in cases where attention pooling would be the more obvious choice, i.e., we only examine the contextual representation of one query or document in isolation.

In addition, the application of LACE (or R-LACE) limits us to only design tasks in which the inputs for our probe model have the same dimension d as the hidden representations of tokens in $\mathcal{M}$. This directly results from the design of LACE (see the formalization in Section 4). So while in *edge-probing*, it is possible to concatenate the pooled representations of a query and a document or two spans and use this as an input

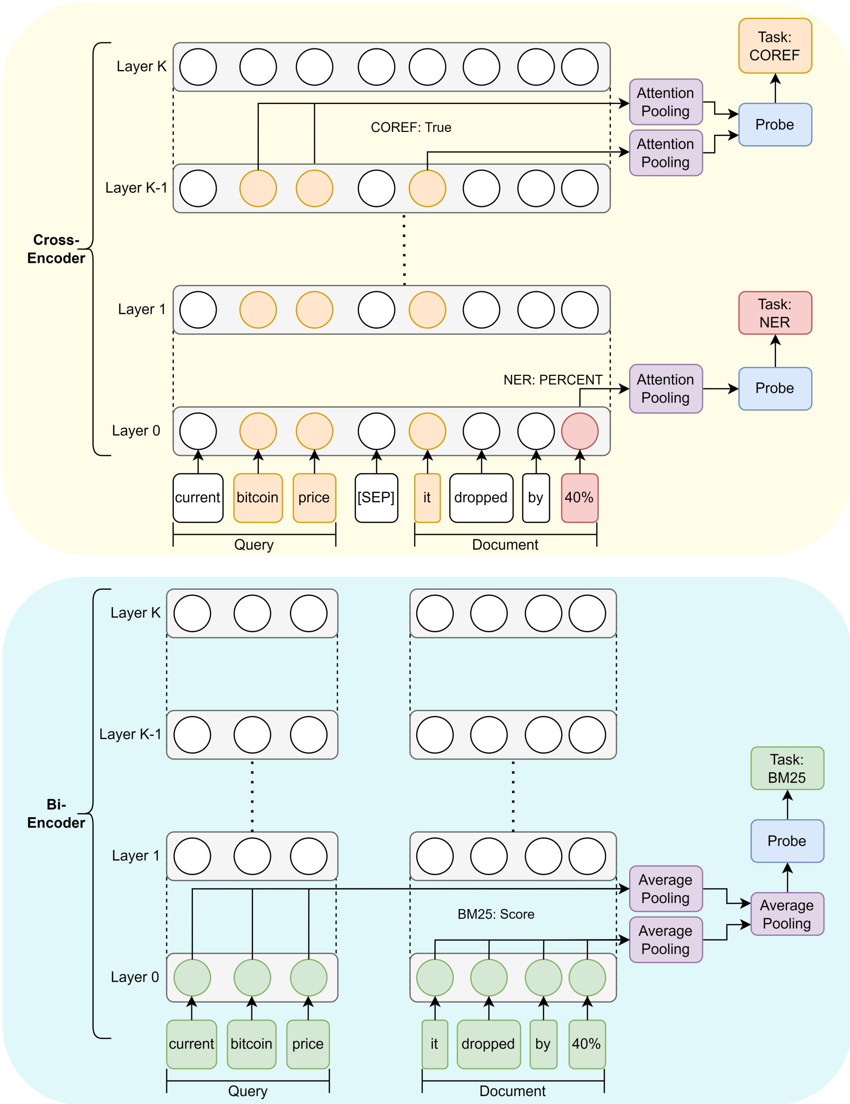

Figure 4.1.: Schematic overview of differences between *edge-probing* a cross-encoder and our approach to probe a bi-encoder. As examples, we show how different probing tasks related to retrieval properties (COREF, NER and BM25) are carried out for a single query-document pair (query: "current bitcoin price", document: "it dropped by 40%").

for a probe, we do not have this option. Instead, we resort to average pooling to combine representations of a query and a document or two spans to the needed dimensionality d.

A schematic overview depicting the described differences of applying *edge-probing* to a cross-encoder and our approach of probing a bi-encoder is in Figure 4.1. The cross-encoder can rely on interactions between the query and the document from the ground up, which is potentially beneficial to determine relevance. The bi-encoder encodes query and document independently from one another and compares their representations afterward.

4.3. Causal Probing Setup

We will now provide a general description of our approach to causally probe the subject model $\mathcal{M}$. The key idea of making the hidden representations of $\mathcal{M}$ oblivious of a property can be seen in Figure 4.2.

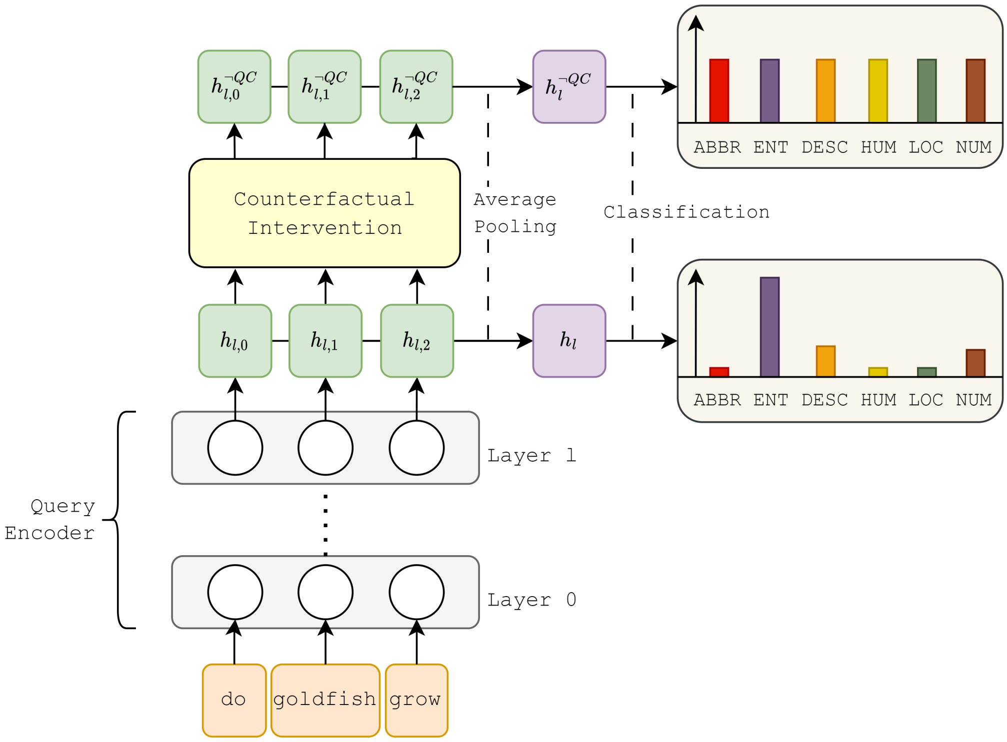

Figure 4.2.: Key idea behind causal probing illustrated by an example of question classification (QC). The model's hidden representations are altered, such that a probe is unable to predict the investigated property, the question type in this case.

First, we construct datasets $D_{\text{TASK},t}$ for tasks t related to retrieval properties. In Section 4.5, we describe how this is done for each property in detail. Samples from a dataset $D_{\text{TASK},t}$ for a task t contain a sequence s, a target value y, and optionally spans sp signaling the position of specific tokens in s (see Table 4.1 for an overview with examples). We distinguish the tasks along two dimensions: Having a regression or a classification target and being a sequence-level or token-level task. In regression, the target values are real numbers, while in classification, they indicate the correct class. In sequence-level tasks, there are no spans. We use the whole sequence, which can either be a query together with a document or only a query or a document. In token-level tasks, we use spans and focus only on specific tokens from the sequence.

Then, we construct a dataset $D_{\text{ENCODED},t,l}$ mirroring the structure of $D_{\text{LACE}} = \{(x_i, y_i)\}_{i=1}^{N}$ for every task t and layer l, containing pooled representations from the subject model $\mathcal{M}$ accompanied with target values, such that we can subsequently apply LACE or R-LACE depending on the task and obtain a projection $P_{t,l}$. Note that this procedure varies conditional on the task characteristics, i.e., whether the task is a sequence-level or token-level task. In Section 4.3.1, we describe the preparatory steps and the application of LACE or R-LACE to obtain $P_{t,l}$ in detail.

To test for the influence of removing linearly present information by projecting with $P_{t,l}$, we first perform an indexing step: We iterate over a corpus of 8.8 million passages (see Section 4.4.1) and do a forward pass through the document encoder of $\mathcal{M}$ up until layer l. The hidden representations at that point are $\{h_{l,i}\}_{i=1}^{T}$ if T is the number of tokens in one passage. Then, we do a counterfactual intervention by projecting the hidden representations with $P_{t,l}$ token-wise resulting in $\{h_{l,i}^{\neg t}\}_{i=1}^{T}$, which should not contain linear information about the property related to task t. Afterward, the forward pass is continued with the remaining layers. As a final operation, the document encoder of our subject model $\mathcal{M}$ performs an average pooling over all token representations to produce the final output vector (see Section 2.3.3), which is then indexed. This procedure is done for every layer l for all passages in the corpus and took approximately 12 hours on our hardware (GTX 1080 Ti) with a batch size of 12.

The index, containing counterfactual representations of passages, enables us to perform the evaluation of the model's retrieval capabilities. We encode a test set of queries and again perform a counterfactual intervention with $P_{t,l}$ in the same manner as described above. The resulting representations are then used to perform a vector similarity search on the counterfactual index to retrieve relevant passages. A schematic depiction of the described procedure is shown in Figure 4.3.

There exist highly sophisticated options to store vectors in an index for efficient similarity retrieval (Johnson et al., 2021). However, we found that it is feasible to use a simple flat index from the *Faiss* library[2] in our case.

To evaluate the effect of the intervention, we incorporate the performance of our subject model $\mathcal{M}$ on an index with all passage representations from the corpus without any intervention as a baseline. In addition, we test the retrieval performance on control representations. The purpose of these representations is to control whether a performance

[2] `https://github.com/facebookresearch/faiss`

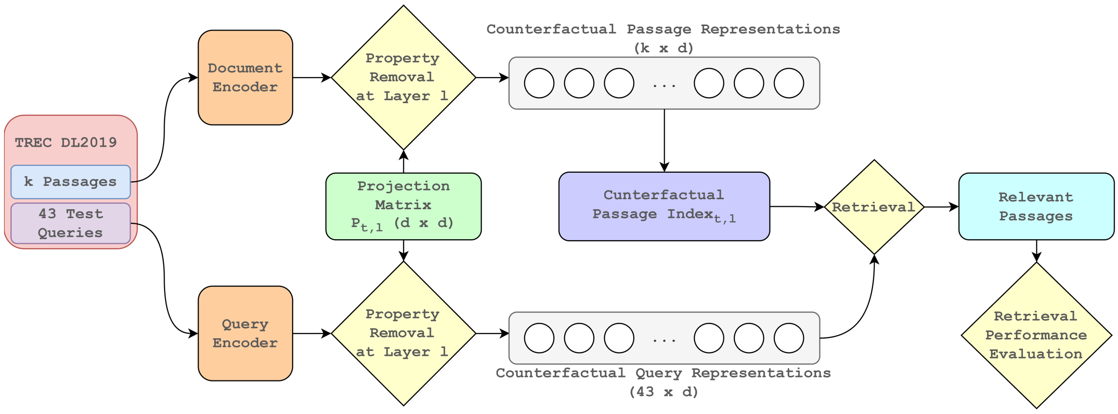

Figure 4.3.: Schematic description of the indexing (upper part) and retrieval & evaluation steps (lower part) to test for the influence of removing a retrieval property approximated by task t at layer l. Note that after removing the property at layer l, the forward pass is continued, which is omitted in this depiction for clarity.

loss can be attributed to the removal of relevant information as opposed to the removal of random information. Elazar et al. (2021) suggested this control task in the investigation of INLP. However, we incorporate a different approach to select and remove random directions. By examining the provided code[3] by Elazar et al. (2021), it came apparent that the removed directions are not necessarily pair-wise linearly independent and do not come from the actual feature space. Therefore, the eliminated subspace does not have rank k when k random directions are removed, which will be necessary for the next feasibility study (Section 5.2). Hence, we want to guarantee to select directions that exist in the feature space and that the removed subspace has a pre-defined rank, which led to the procedure described in Alg. 1.

As an approximation of the feature space, we sample 50 000 passages from the corpus and perform a forward pass with our subject model $\mathcal{M}$ until layer l. Then, we use the resulting representations as the input for Alg. 1 to obtain a projection that randomly removes a subspace from the representation space at layer l. To evaluate the model on control representations, we perform the same procedure as depicted in Figure 4.3, but with the control projection.

[3] `https://github.com/yanaiela/amnesic_probing/blob/120de7e63b6d2332da8859ff90d4340dbfda4ab6/amnesic_probing/encoders/encode_with_forward_pass/encode.py`

Algorithm 1: Random removal of a subspace of rank k

```
Input: Rank of the removed subspace: k ≥ 1, feature matrix: X ∈ ℝ^{N×d}
B ← orthonormal basis(X^⊤) ;
rb ← rank(B);
/* Cap rank of subspace to be eliminated (k) if it exceeds rank of
   the whole space.                                                  */
if k > rb then
|  k ← rb
end
/* Randomly choose k basis vectors B_i from B                        */
ri ← random choice(rb, k);
rp ← empty list;
foreach i ∈ ri do
|  rp.append(B_i × B_i^⊤)
end
/* Get projection to subspace of rank rb − k by subtracting the
   summed up random projections from the identity matrix I_d         */
P ← I_d − sum(rp, axis = 0);
return P
```

4.3.1. Property Removal

This section explains how we obtain a projection $P_{t,l}$ for every task t and layer l. First, we need to prepare a dataset mirroring the structure of $D_{\text{LACE}} = \{(x_i, y_i)\}_{i=1}^{N}$. To do so, we use our subject model $\mathcal{M}$ and a task dataset $D_{\text{TASK},t}$. We construct a dataset called $D_{\text{ENCODED},t,l}$ for every layer l. The target values y_i are simply copied from $D_{\text{TASK},t}$. To obtain x_i, we distinguish by task characteristics.

For sequence-level tasks, we do the following: Let T be the number of tokens in a sequence s from $D_{\text{TASK},t}$. We perform a forward pass of s through $\mathcal{M}$. If s contains a query *and* a passage, we perform two separate forward passes. If it only contains a query *or* a passage, we perform a single forward pass. Then, we average over the fixed hidden representations $\{h_{l,i}\}_{i=1}^{T}$ with $h_{l,i} \in \mathbb{R}^d$ at layer l token-wise. The resulting pooled vector is $x_i \in \mathbb{R}^d$.

For token-level tasks, we perform a forward pass like above, but instead of pooling over all the hidden representations of tokens, we only pool over the ones specified by the spans sp.

Now, if our task is a regression problem, we can directly retrieve $P_{t,l}$ by using the closed form solution of LACE (see Section 4.1). In the other case, we run R-LACE with the predictor being a logistic regression model. More specifically, a binary logistic regression model if there are only two classes, and a multinomial logistic regression model if there are more than two classes (see Section 4.1.2). The training of the predictor in one iteration of the minimax game is carried out with stochastic gradient descent. The learning rate schedule is calculated by

$$\eta = \frac{1}{\alpha(t + t_0)} \tag{4.5}$$

with t being a learning step, t_0 is chosen according to a heuristic by Bottou (2012) and $\alpha = 1\text{e}{-4}$. α also serves as the weight of an $L2$ parameter regularization. The learning stops early if the loss improves less than a tolerance of 1e−4 after 15 iterations. A maximum of 25 000 epochs are trained. The stopping criterion of the whole minimax game is met when the predictor only achieves majority-accuracy with a tolerance of 2e−3 or a maximum of 50 000 iterations have been reached. For further details on the procedure of R-LACE, we refer the reader to Ravfogel et al. (2022a). We obtain the wanted projection $P_{t,l}$ as soon as the minimax game terminates.

4.4. External Datasets

In this section, we name and describe the external datasets we used to automatically generate[4] the task datasets $D_{\text{TASK},t}$ for each property. The first external dataset is from the TREC 2019 Deep Learning Track (Craswell et al., 2020). The second one is a dataset for question classification (Li and Roth, 2002).

4.4.1. TREC 2019 Deep Learning Track Dataset

There exist two tasks in the track, each with its individual dataset: One with a focus on document retrieval and the other on passage retrieval. Essentially passages are short sections from a document. Each task has sub-tasks composed of a full retrieval and a re-ranking of top documents or passages retrieved by a simple first-stage retrieval system, such as BM25. Since our subject model $\mathcal{M}$ prefers shorter input sequences, we use only the dataset for the full passage retrieval task.

The datasets originate from the MSMARCO dataset (Nguyen et al., 2016), whose goal is to contribute to the development of a question-answering system used in automated speech or text assistants. The questions are collected from real-world query logs of the search engine Bing. To create the corresponding answers, they retrieve relevant documents from Bing's web index and extract passages containing relevant information in an automatic manner. They do not provide any details on this process but have the passages and questions subsequently reviewed by crowd workers, whose task is to annotate how appropriate the passages are in light of the given question.

[4]The code, for the automatic task dataset generation, can be found here: `https://github.com/Heyjuke58/tinse_probing_datasets`

The TREC 2019 passage retrieval corpus[5] contains roughly 8.8 million passages. The test set contains 43 queries with corresponding relevance labels for passages. In addition, there exist 6.6 thousand queries with associated top-1 000 passages[6], from the validation set of the TREC 2019 passage retrieval dataset, which will be useful to us in the automatic generation of task datasets $D_{\text{TASK},t}$.

From now on, we will use the term *document* if we speak in the context of general IR and *passage* if we speak about our specific experimental setup. However, in this work, *passages* are the *documents* from the IR perspective.

4.4.2. UIUC's CogComp Question Classification Dataset

This dataset is provided by Li and Roth (2002)[7] and contains 5 453 questions with their corresponding question type. They include a coarse and a fine-grained taxonomy. We will use the coarse classes, which include ABBREVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION, and NUMERIC VALUE. In contrast to the TREC 2019 passage retrieval dataset, we can directly use this dataset as one of our task datasets $D_{\text{TASK},t}$.

4.5. Retrieval Properties

In this section, we elaborate on the properties we will examine and explain our intentions to do so. Neural IR models optimize for relevance. However, it is not an easy task to pinpoint what a document needs to generally fulfill to be relevant to a query. While classical retrieval approaches focus on query terms being matched in documents, neural IR provides the ability to semantically compare queries and documents, opening a wide variety of possible explanations of how to determine relevance. Therefore, our approach is to look at this problem from different angles, reflected in the choice of properties we investigate.

Furthermore, we describe the generation process of the associated dataset $D_{\text{TASK},t}$.

4.5.1. BM25

We already explained how BM25 works in Section 2.1.2. Since it is a solid exact term-matching retrieval method very capable of performing first-stage retrieval, removing it could give us interesting insights into whether our studied bi-encoder relies on such a simple mechanism. As our model only performs a late interaction (see Section 2.3.2), it would need to encode the term frequency of document terms since it could not rely on

[5]TREC 2019 datasets: `https://microsoft.github.io/msmarco/TREC-Deep-Learning-2019`

[6]While the majority (94%) of queries have exactly 1000 relevant passages associated with them, some queries have way less than that, going down to only one passage. On average, each query has 955.

[7]`https://cogcomp.seas.upenn.edu/Data/QA/QC/`

Task	Type	Level	Example
BM25	Regression	Sequence	**query**: most expensive hotels in new york city **passage**: The world's most expensive flight costs $38,000 — one way. Etihad Airways' new route connecting Mumbai and New York City... **target**: 22.063
SEM	Regression	Sequence	**query**: does insulin give you constipation **passage**: Summary: Constipation is found among people who take Insulin, especially... **target**: 0.132
AVG TI	Regression	Sequence	**query**: how long can ribs stay frozen **passage**: Raw pork chops can be safely frozen for up to six months... **target**: 2.763
TI	Regression	Token	**query**: where is hamvir's rest in skyrim **passage**: Hearthfire is the second DLC release for [**Skyrim**] behind the extremely successful... **target**: 10.336
NER	Classification	Token	**passage**: If you want to meet halfway between [**Los Angeles**], CA and Stockton, CA or just... **target**: Geopolitical entity (GPE)
COREF	Classification	Token	**passage**: [**Aluminum chloride**] is a chemical compound that has several uses, including as a treatment for excessive sweating and in antiperspirants. [**It**] is used... **target**: True
QC	Classification	Sequence	**query**: What is the full form of .com? **target**: Abbreviation (ABBR)

Table 4.1.: Overview of designed tasks accompanied by an example sample from the respective dataset $D_{\text{TASK},t}$. Samples from $D_{\text{TASK},t}$ contain a target, a sequence, which can either be a query-passage pair or a passage or a query, and optionally one or two spans if the task is token-level. The square brackets indicate which words are contained in the spans.

knowing which terms appear in the query. Also, to encode IDF information, it would need to learn and remember corpus-level statistics.

To build the task dataset, we sample queries from the validation set of the TREC 2019 passage retrieval dataset, as well as passages from the top-1 000 of the sampled queries and calculate the BM25 scores (see Section 4.5.1) of the in total 60 000 query-passage pairs. The BM25 calculation is carried out via the Elasticsearch BM25 implementation[8].

4.5.2. Semantic Similarity (SEM)

In contrast to exact term-matching, it could be interesting to investigate if our model relies on soft term-matching to determine relevance, i.e., looks for semantically familiar terms. We attempt to isolate this concept by defining a semantic similarity score between a query and a document. To quantify the semantic similarity, we embed non-stop-word terms from the query and the document in the *GloVe* (Pennington et al., 2014) embedding space. Then, we average over the sequence dimension query-wise and document-wise. Lastly, we take the cosine similarity between the two resulting vectors, corresponding to the query and the document.

To build the dataset, we use the same sampled queries and passages as for BM25 and proceed with the calculation of SEM scores as described above.

4.5.3. (Average) Term Importance ((AVG) TI))

Formal et al. (2022) investigate whether neural IR models for single-stage retrieval perform lexical matching in an appropriate manner, i.e., for important terms w.r.t. the query, on out-of-domain corpora. They use the Robertson-Spärck-Jones (RSJ) weight (Robertson et al., 1976) for a term t as their measure of term importance w.r.t. to a query q and the corpus $\mathcal{C}$. It calculates as follows:

$$RSJ(t, q, \mathcal{C}) = \log \frac{p(t|\mathcal{R})p(\neg t|\neg\mathcal{R})}{p(\neg t|\mathcal{R})p(t|\neg\mathcal{R})} \tag{4.6}$$

where $p(t|R)$ is the probability that t occurs in relevant documents $\mathcal{R}$ and $p(\neg t|\neg\mathcal{R})$ is the probability that t does not occur in non-relevant documents $\neg\mathcal{R}$. The probabilities from the denominator are defined accordingly. To compute the weight, we need relevance labels for the passages from the corpus for each query to know which passages belong to the relevant or non-relevant set, respectively. To formulate a probing task, we test two approaches: First, we average over all non-stop-word terms from a passage and try to predict this value. Second, we try to predict the term importance of specific passage terms. Note that term importance is always defined w.r.t. a query.

To build the dataset, we use the same sampled queries and passages as for BM25 and SEM. For every query-passage pair, we tokenize the passage and remove stop-words. Then, we calculate the IDF values for the relevant passages w.r.t. the query by considering the top-1 000 passages for the query as the relevant passages. Note that according to Craswell et al. (2020), the top-1 000 passages are retrieved via BM25, which is somewhat problematic since BM25 is not a gold standard for relevance. However, in the

[8] `https://www.elastic.co/elasticsearch/`. Wrapper library we used: `https://github.com/kwang2049/easy-elasticsearch`.

absence of more accurate relevance labels, we resort to this option. In addition, we calculate IDF values for the whole corpus of 8.8 million passages and restrain from cutting out the top-1 000 relevant passages from it for time budget reasons. The two sets of IDF values give us the ability to calculate the RSJ weight for terms. For the dataset concerning AVG TI, we average over the RSJ weight for all non-stop-word tokens from the passage. For the TI dataset, we rank the terms in descending order by their RSJ weight and select four terms with their corresponding span and RSJ weight as a target to include in the dataset: The first, the last, the one leading the second quarter and the one leading the third quarter.

One has to mention that term importance, as we design it, is highly related to IDF. The probabilities of terms occurring or not occurring can be directly extracted from IDF by taking the document frequency and dividing it by the total number of passages. However, as opposed to BM25, the term importance is now defined w.r.t. to a query. This forces our bi-encoder to not only encode which terms are contained in a passage but also in which contexts they are important.

4.5.4. Named-Entity Recognition (NER)

As opposed to the three previously described properties, which all to some degree rely on hard or soft term-matching, named-entity recognition aims in a quite different direction. NER is a subtask of NLP and consists of identifying and categorizing entities in a text. The entity classes must be pre-defined. We use the *spaCy* library (Honnibal et al., 2020), which defines 18 classes, ranging from persons, companies, and events to dates, monetary values, and weight or distance measurements, to name a few. To understand the semantics of a document, which we think could be highly important to predict its relevance to a query, the capability of NER plays a key role. To put our bi-encoder to the test, we do not need a combination of a query and a passage, as it needs to be capable of encoding information about named entities in the representations of a passage or a query in isolation.

Therefore, we only sample passages from the corpus. Next, we use the *spaCy* library[9] to perform the task of NER on the sampled passages. As this is likely not 100% accurate, there will exist errors in our dataset. We sample two of the found entities to include in the dataset. If a passage does not contain at least two named entities, we continue with another sampled passage and leave the former out of the dataset. We repeat this until our dataset reaches 60 000 passages.

4.5.5. Coreference Resolution (COREF)

Equally, as NER, coreference resolution is a subtask of NLP. A coreference occurs when two or more expressions refer to the same entity. For a language model, it is important

[9] `https://spacy.io/`

to detect them, in order to correctly encode the semantics of a sentence or a group of sentences. While this holds true when we consider only one document in isolation, it becomes more complicated when a coreference appears across a query-document pair. A bi-encoder cannot rely on using cross-attention between the query and a document. It could encode information about the entity referenced in both the query and the document, which would in turn help determine relevance. When we try to eliminate the capability of the model to detect if terms refer to the same entity, we might damage its potential practice to encode mentions of the same entity similarly.

To build this dataset, we use the *spaCy* library in combination with the pipeline extension *neuralcoref*[10]. It allows us to automatically detect coreferences. We randomly sample passages from the corpus of 8.8 million passages and add them to the dataset until 60 000 passages are reached. A passage is accepted if it contains a coreference, and we can construct a negative example for this coreference. The negative example is either a random word or another entity from the passage that is not part of the coreference.

4.5.6. Question Classification (QC)

A bi-encoder could potentially categorize documents by the type of information they provide and queries by the information need they convey. If a document is known to answer an information need of a particular format, such as the date of a historical event, relevance to a query that asks for a date could be influenced by that. To test if our studied bi-encoder behaves in that way, we try to limit the model's capability to encode the type of question and, therefore, the type of information need that is conveyed by the query.

The sequences in the task dataset are solely queries, more precisely natural language questions, directly taken from the external dataset (see Section 4.4.2) together with their classification target. Users tend to prompt keyword searches instead of natural language questions, which might reduce the usefulness for the IR system to rely on a classification of the information need. Keyword searches could be more difficult to classify due to ambiguity. As we only test our retrieval model on 43 hand-picked test queries, we can at least estimate whether we face this problem in our setup. But generally speaking IR models might not be able to take full advantage of this property.

By examining the 43 test queries, we can confirm that 25 are indeed natural language questions, only missing a question mark. The 18 remaining queries can all be transformed into a natural language question by adding stop-words. For example, the keyword query "rsa definition key" could be transformed into "How is a key defined in rsa?". However, since RSA produces a public-private key pair, some ambiguity concerning the user's information need remains. This example is one of the more critical of the 18 non-natural language questions. Others, like "definition declaratory judgment" or "cost of interior

[10]`https://spacy.io/universe/project/neuralcoref`

concrete flooring" are more obviously almost natural language questions. Boiled down, we are confident that the classification of these queries could be beneficial for our subject model $\mathcal{M}$ to determine the relevance of passages.

As this task solely focuses on queries, we do not intervene on the passages that form the index for the retrieval evaluation, as described in Section 4.3 and Figure 4.3. We use a passage index with unaltered representations and only intervene on the query representations.

4.5.7. Other Possible Properties

We want to list other properties that might be interesting to investigate but did not make it into this work.

Semantic Role Labeling

In semantic role labeling (Palmer et al., 2010), the goal is to fill slots in a conceptual frame with words or phrases from a text that take specific semantic roles in the frame. In the sentence "John helps Mary by taking some tasks off her hands at work." the agent role is filled by the helper "John" while "Mary" is the patient, the helpee. Successfully solving this task indicates a semantic understanding of the text, which can in turn be helpful when prompted with an information need related to the text.

Semantic Proto-Role Labeling

As an extension of the previous task, semantic proto-role labeling assigning fine-grained non-exclusive semantic attributes to arguments. E.g., the argument "John", who fills the agent role, could be assigned the attribute *volitional involvement* since he presumably helps "Mary" voluntarily (Reisinger et al., 2015).

Named-Entity Linking

While in NER, the goal is to classify an entity into pre-defined categories, one can also link a detected entity to the exact match in a knowledge base like *DBpedia*[11]. This would require the system to have learned world knowledge and disambiguate the entity from others depending on the context. For example, to decide whether "Walter White" refers to a character in a television show or a former leader of the *National Association for the Advancement of Colored People*. Undoubtedly, the correct or erroneous labeling, in this case, would have an impact on retrieving relevant documents. (Kolitsas et al., 2018)

[11] https://www.dbpedia.org/

4.6. Evaluation Metrics

This section will state the evaluation metrics we use in our experiments.

4.6.1. Probing

We conventionally probe our subject model $\mathcal{M}$ in feasibility studies described in Chapter 5. For this purpose, we use the following evaluation metrics.

Accuracy

Accuracy is an evaluation metric for classification. It measures the proportion of correct predictions to the total number of predictions. Let y be the true labels and $\hat{y}$ the predicted labels.

$$\text{Accuracy}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \begin{cases} 0, & \text{if } y_i \neq \hat{y}_i \\ 1, & \text{if } y_i = \hat{y}_i \end{cases} \tag{4.7}$$

Coefficient of Determination (R^2)

The coefficient of determination, denoted as R^2, is an evaluation metric for regression analysis. It shows the proportion of the variation in the dependent variable that is predictable from the independent variables. Given a dataset with feature vectors $\{(x_1, ..., x_k)_i\}_{i=1}^n$ and corresponding target values $\{y_i\}_{i=1}^n$ we can fit a linear regression model $f_i = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k$ by finding appropriate coefficients β, such that f models the relationship between the feature vectors x and the target values y. R^2 calculates as follows:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}, \tag{4.8}$$

where $SS_{res} = \sum_i e_i^2$ is the sum of squares of residuals and $SS_{tot} = \sum_i (\bar{y} - y_i)^2$ is the total sum of squares. The residual $e_i = y_i - f_i$ is the difference between the true value y_i and the predicted value f_i. $\bar{y} = \frac{1}{n} \sum_i y_i$ is the mean of the target values. Usually, R^2 yields values in the range from 0 to 1, where a value of 1 denotes a perfect fit of the regression model to the data. But, the values can also be negative. In this case, the mean $\bar{y}$ would better model the relationship than the fitted predictor f. Compared to another common evaluation metric for regression analysis, the mean squared error (MSE), R^2 allows for easier comparisons across datasets because of the scaling to the total sum of squares.

4.6.2. Retrieval

For the evaluation of retrieval performance, we use the test set of 43 queries from the TREC 2019 Deep Learning Track. All passages have assigned labels y_i w.r.t. the test queries on a four-point scale from 3 (perfectly relevant) to 0 (not relevant).

We will now explain what the Normalized Discounted Cumulative Gain (NDCG) measures and how it can be computed for a retrieved list of documents given the ideal ranking order. As it is the main metric in the TREC 2019 Deep Learning Track and a popular choice for retrieval in the context of web search, we will use it to measure the retrieval performance of our subject model $\mathcal{M}$ after counterfactual interventions.

NDCG

NDCG is calculated as the proportion of Discounted Cumulative Gain (DCG) and the ideal DCG.

$$\mathrm{NDCG} = \frac{\mathrm{DCG}}{\mathrm{IDCG}} \tag{4.9}$$

The intuition of DCG is to logarithmically scale the gain of retrieved documents depending on their position in the ranking. The gain corresponds to their relevance label y_i ranging from 3 to 0. It penalizes ranking highly relevant documents at later positions. The following formula describes how it is calculated:

$$\mathrm{DCG} = \sum_{i=1}^{|\mathcal{C}|} \frac{y_i}{\log_2(i+1)} \tag{4.10}$$

where $|\mathcal{C}|$ is the total number of documents in the corpus.

The ideal DCG is calculated by ordering the relevance labels descendingly and measuring the DCG. Note that there may exist multiple optimal orderings. Therefore, the best value for NDCG is 1, and it is achieved when the ranking is one of the optimal ones. As a ranking of all documents is often not reasonable, the NDCG can be calculated for the first k ranked documents, indicated by an appended @k.

5. Feasibility Studies

Our main experiment, framed as *causal probing*, investigates how the removal of certain properties influences the retrieval behavior of our subject model $\mathcal{M}$. However, we saw the need to take a closer look at the *removal* itself. Therefore, a large portion of this work focuses on validating whether it is justified to derive causal explanations in our main experiment by investigating how our used property removal technique LACE operates. These preliminary experiments are referred to here as feasibility studies.

5.1. Probing as a Sanity Check

As Elazar et al. (2021) pointed out, removing properties in an attempt to causally explain their influence on a model is only reasonable if the property can actually be predicted from the representations. Otherwise, the model would not be able to use the information at all. Therefore, as a sanity check, we employ conventional probing to test if we can predict our investigated properties with linearly present information. Furthermore, we want to make sure it is indeed difficult for a linear model, or impossible in case of linear regression (see Section 4.1.1), to predict the property after it has been removed by LACE. This serves as an additional sanity check to test whether the removal technique behaves as we would expect. The removal would be considered successful if the probe performs close to majority accuracy. For ease of comparison, we reframe the regression tasks (BM25, SEM, TI, AVG TI) as classification tasks by binning their targets into 10 equally sized bins and test both of the versions[1].

Additionally, we repeat both sanity checks with a non-linear probe to test how the removal of linearly present information affects the more expressive probe and get an impression of what a powerful probe can achieve on our tasks. Note that there are reasons to avoid probes of high expressiveness when trying to discover the information a language model learns, as they might learn the task themselves, without exhaustively making use of the encoded information (Hewitt and Liang, 2019; Pimentel et al., 2020). However, since our goal is not to draw conclusions about the distribution of encoded knowledge regarding the investigated properties across layers, we do not need to be too cautious regarding the choice of our non-linear probe.

[1]In the binned versions we still use the projection matrix we obtained from the closed form solution (regression setting, LACE, see Section 4.1.1), as it did not make a difference to intervene with the projection obtained from the relaxed minimax game (classification setting, R-LACE, see Section 4.1.2)

Accordingly, we are measuring the performance of a linear and a non-linear model predicting the property trained on the following data:

1. **Original representations**
2. **Counterfactual representations** (information about the property removed by LACE)
3. **Control representations** (same number of directions removed as in the counterfactual representations, but randomly)

For each type of representation, we will now describe the preparatory steps we perform to obtain them.

Original Representations: In Section 4.3.1 we described how to obtain $D_{\text{ENCODED},t,l} = \{(x_i, y_i)\}_{i=1}^N$ for a task dataset $D_{\text{TASK},t}$. The feature vectors $x_i \in \mathbb{R}^d$ from $D_{\text{ENCODED},t,l}$ are what we here call the original representations and y_i are the targets. Depending on the task those are either real values, binary values, or one-hot vectors. Note that we also retrieve the representations for the test set of our task datasets, as we will evaluate the probes with them.

Counterfactual representations: Before we pool over the hidden representations of tokens $\{h_{l,i}\}_{i=1}^T$, to obtain a d-dimensional vector like described in Section 4.3.1, we alter them by projecting token-wise with $P_{t,l}$. This gives us new representations $\{h_{l,i}^{\neg t}\}_{i=1}^T$, which should not contain any linearly present information about the property described by task t. Those are then average pooled, either all or only the tokens from the spans sp, to obtain the counterfactual representations.

Control representations: We obtain these in the same way as above, except that we instead apply a projection that removes a subspace of the same rank as in the counterfactual representations, but one chosen at random. We use Algorithm 1 with the feature data of the specific dataset as the input for the algorithm.

We repeat all the experiments for 5 runs with different random seeds and report the mean and 95% confidence intervals.

We will now describe the probe models we use for this experiment and how they are trained. Remember that we need probes for regression and classification tasks. Concerning the evaluation of our probes, accuracy is used for the classifiers, and R^2 is used for the regressors (see Section 4.6.1 for the description of the metrics).

Linear Probes

For the regression tasks we use a ridge regression model (Hilt and Seegrist, 1977) $f(x) = xw$ with coefficients $w \in \mathbb{R}^{d\times 1}$, which minimizes the following objective:

$$\min_w ||y - Xw||_2^2 - \alpha||w||_2^2 \tag{5.1}$$

α weights the regularization term for size of the coefficients w and is set to $\alpha = 1$. The coefficients are calculated by applying singular value decomposition (SVD).

For the classification tasks, we use a multinomial or binary logistic regression, depending on the task at hand. In binary logistic regression the binary cross-entropy loss over all samples $\{x_i, y_i\}_{i=1}^N$ is minimized:

$$\min_{w,b} -\frac{1}{N}\sum_{i=1}^{N} y_i \log \sigma(x_i w + b) + (1 - y_i)\log(1 - \sigma(x_i w + b)) \tag{5.2}$$

with σ being the sigmoid function $\sigma(x_i w + b) = \frac{1}{1+\exp(-(x_i w + b))}$ and $w \in \mathbb{R}^d$, $b \in \mathbb{R}$ being the learned parameters.

In multinomial logistic regression the cross-entropy loss is minimized:

$$\min_{w,b} -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k} \log \frac{\exp(x_i w^{(k)} + b^{(k)})}{\sum_{j=1}^{K}\exp(x_i w^{(j)} + b^{(j)})} \tag{5.3}$$

with K classes and $w \in \mathbb{R}^{d\times K}$, $b \in \mathbb{R}^K$ being the learned parameters. $w^{(i)}$ is the i-th column of w and $b^{(i)}$ the i-th entry of b.

The training resembles the already described in Section 4.3.1, except that the early stopping condition has a patience of 10 iterations and is based on the performance on a validation set, built by sampling 10% from the training set. Also, a maximum of 3 000 epochs are trained.

Non-Linear Probes

As the non-linear model, we use a comparably more complex 3-layer MLP, in contrast to the probes used by Tenney et al. (2019b) or Hewitt and Liang (2019), with ReLU activation and a hidden layer size of $m = 100$:

$$f(x) = \mathrm{ReLU}(\mathrm{ReLU}(xW^{(0)} + b^{(0)})W^{(1)} + b^{(1)})W^{(2)} + b^{(2)} \tag{5.4}$$

where $W^{(0)} \in \mathbb{R}^{d\times m}$, $W^{(1)} \in \mathbb{R}^{m\times m}$, $W^{(2)} \in \mathbb{R}^{m\times c}$ and $b^{(0)}, b^{(1)} \in \mathbb{R}^m$, $b^{(2)} \in \mathbb{R}^c$. In the regression setting the MLP acts as a regressor ($c = 1$), and in the classification setting it acts as a classifier with c target classes. Training is conducted with an initial learning rate of 1e−3, *Adam* optimizer (Kingma and Ba, 2015), and a batch size of 200. A maximum of 300 epochs are trained.

5.2. Eliminating Subspaces of Increasing Ranks

In the experiments of Ravfogel et al. (2022a), they solely considered binary classification and found that the elimination of rank-1 subspace via R-LACE suffices to make a predictor oblivious of the concept. To see whether this also holds for the tasks we designed in this work, including multiclass classification, we conduct this feasibility study.

The obvious choices of tasks for this experiment are NER, being an 18-fold classification, as well as QC, being a 6-fold classification. We apply R-LACE on the datasets $D_{\text{LACE},t,l}$ for task t and layer l with ten logarithmically scaled choices of k from 1 to 630. For time budget reasons we limited the maximum iterations of R-LACE to 20 000, which is 40% of the budget compared to the experiment in 5.1. The obtained projections $P_{t,l}$ are subsequently used to alter the feature data from $D_{\text{LACE},t,l}$ and train the same linear probe as in Section 5.1 to solve the task. In addition, we train the linear probe on control representations with the same number of directions removed, but random ones. We use the algorithm stated in Alg. 1 to remove random directions with the feature data of the task datasets as the input for the algorithm. Then, we compare the resulting accuracy with the original accuracy the linear probe achieved, without an intervention on the representations, and calculate the proportional performance drop.

We expect R-LACE to be more efficient in terms of the probe not predicting better than majority accuracy, i.e. only needs to eliminate subspaces of lower ranks, compared to the control representations. But, with increasing subspace rank, the probe should also struggle to solve the task on the control representations.

5.3. Results

We will now present and discuss the results of the two previously described feasibility studies. We will lead with the results of the experiment from Section 5.2, since hyperparameter choices for R-LACE in the experiment from Section 5.1 depend on it. After that, we will check if the sanity checks for each designed task related to a property can be considered passed and discuss the general findings of the probing and the counterfactual intervention.

Results of Eliminating Subspace of Increasing Ranks

From looking at Figure 5.1 our expectation from Section 5.2 can be loosely confirmed. We see that R-LACE needs a substantially lower rank of the eliminated subspace to remove the concept compared to the control representations. Likewise, although only at high ranks, the performance degrades for the control representations.

Most importantly, we see that in both cases (NER and QC) the removal of a subspace of rank 1 does not suffice to make the predictor oblivious of the property. For NER a good choice of the eliminated subspace rank seems to be $k = 8$ and for QC $k = 4$.

Ethayarajh (2019) showed that the geometry of contextualized word representations in BERT is far from being isotropic, occupying only a narrow cone in the vector space. This circumstance gets more prominent in higher layers with a peak in narrowness at layer 11. In part, this could explain why for NER we see a higher proportional accuracy drop for later layers, especially for layer 11. However, when looking at the control heatmap for QC we cannot confirm this observation. In fact, we see the opposite at layer 11,

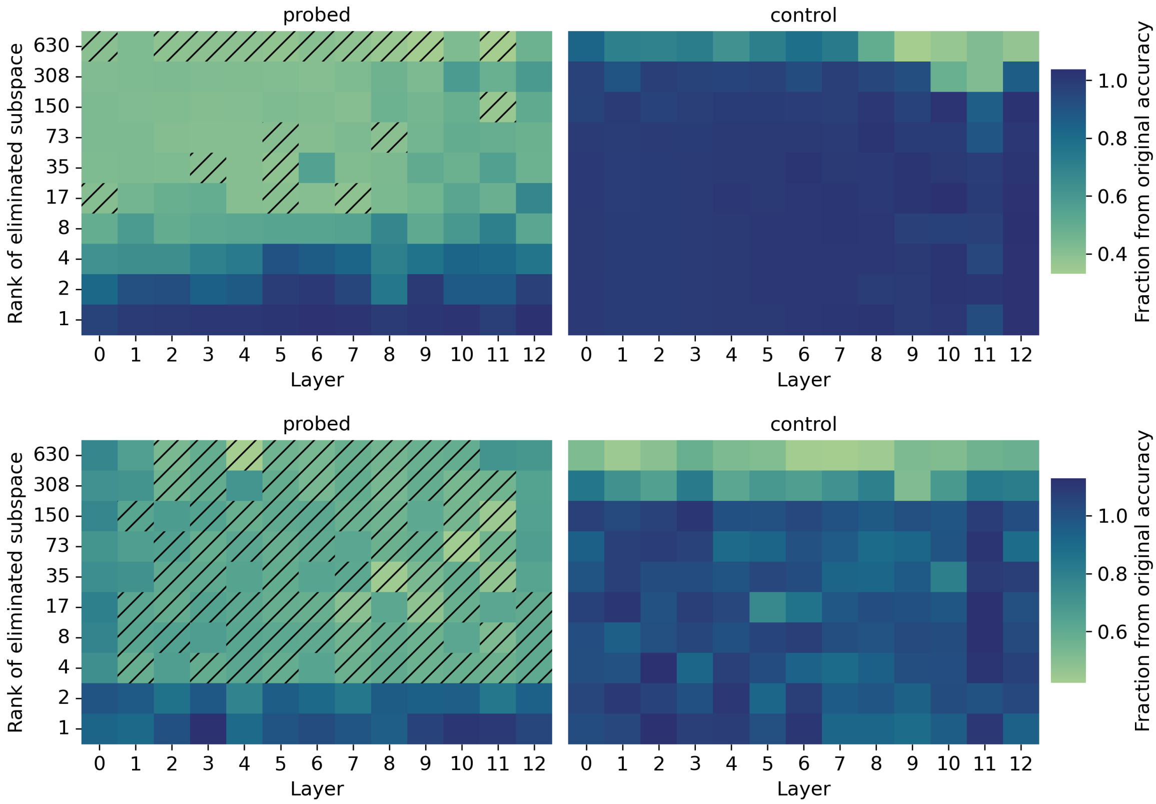

Figure 5.1.: Top row: NER. Bottom row: QC. The heatmaps visualize the evolution of the proportion of the accuracy on the counterfactual or control representations to the original accuracy at each layer with increasing rank of the eliminated subspace. A value of 1 indicates no accuracy decrease. The stripes indicate where the accuracy is equal or less than the majority accuracy (0.24 for NER, 0.23 for QC).

compared to NER. But, when looking at Figure 5.4 this observation does not seem that surprising after all, since the performance on the original representations has a drop at layer 11, which is somehow not the case for the control representations.

In essence, the hyperparameter k controls a trade-off: The higher the k, the more likely it is to remove all linearly present information about the property. However, as k increases, the selectivity of what is removed might degrade, affecting information not related to the property. Therefore, a task-dependent thorough investigation of how k should be chosen is advisable.

Result of Sanity Checks

Now we will present and discuss the results from the feasibility study from Section 5.1. First, we discuss whether the sanity checks for each individual task can be considered passed until we move on to share general findings shared across tasks.

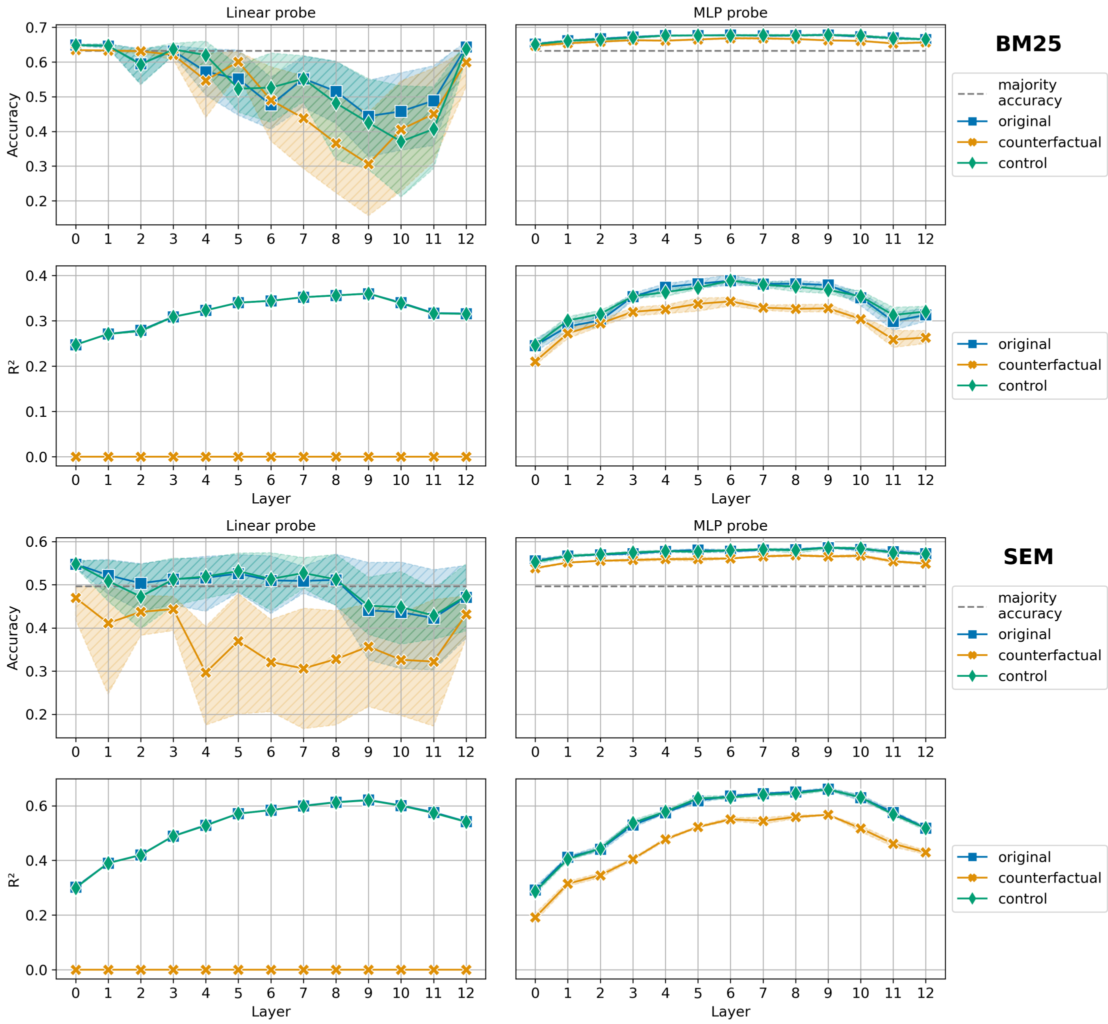

Figure 5.2.: Results of sanity checks for tasks BM25 & SEM. The top two rows depict BM25 and the bottom two SEM. Each individual top row: Binned version (classification). Each individual bottom row: Regression version.

BM25

The top two rows of Figure 5.2 depict the results for BM25. We can see that the linear probe in the binned case does not achieve an accuracy higher than majority, implying that the property is not linearly predictable from the representations. The counterfactual intervention (yellow curve) is nevertheless capable of further degrading performance on the task, especially after layer 6. We observe a considerable amount of variance in the results, which indicates an instability in adapting to the task and removing the property. Based on these findings, it does not seem reasonable to believe that we are able to remove the concept of BM25 from the representations. Nonetheless, we will continue with the causal probing for this task. One possible reason we get these results could be that the information is mostly encoded non-linearly, except for the last layer where no non-linearity follows, which also explains the performance gain of the linear probe. When we look at the accuracy of the MLP we see that it is at least slightly above majority accuracy. Another reason could be that our subject model $\mathcal{M}$ does not encode BM25, at least not how we designed the task. However, according to Choi et al. (2022) information about IDF, which is an integral part of BM25, can be found in the representations of BERT.

Semantic Similarity (SEM)

Moving on to SEM (bottom two rows in Figure 5.2), we observe a similar situation as for BM25, although not as obvious. Our linear model learned on the original representations in the binned case is only barely able to predict SEM on average above majority accuracy until layer 8. After that, the accuracy drops below it on average, which could be accounted to higher layers (9-11) mostly handling higher level semantics (Peters et al., 2018b), triggering the need for more complex non-linear relationships. Unlike BM25, the amnesic intervention has a noticeable effect on average throughout all layers, but we again observe a large amount of variance in accuracy. Taking all points into consideration, we see limitations in the evidential value of subsequent experiments for this task, as for BM25.

Average Term Importance (AVG TI)

Next in line is AVG TI in Figure 5.3 (top two rows). All the mentioned points for BM25 apply here as well. We suspect our task definition to be part of the problem that the property cannot be predicted. The average term importance over all non-stop-word terms (Section 4.5.3) might not be descriptive enough for the predictor to comprehend the concept of this property.

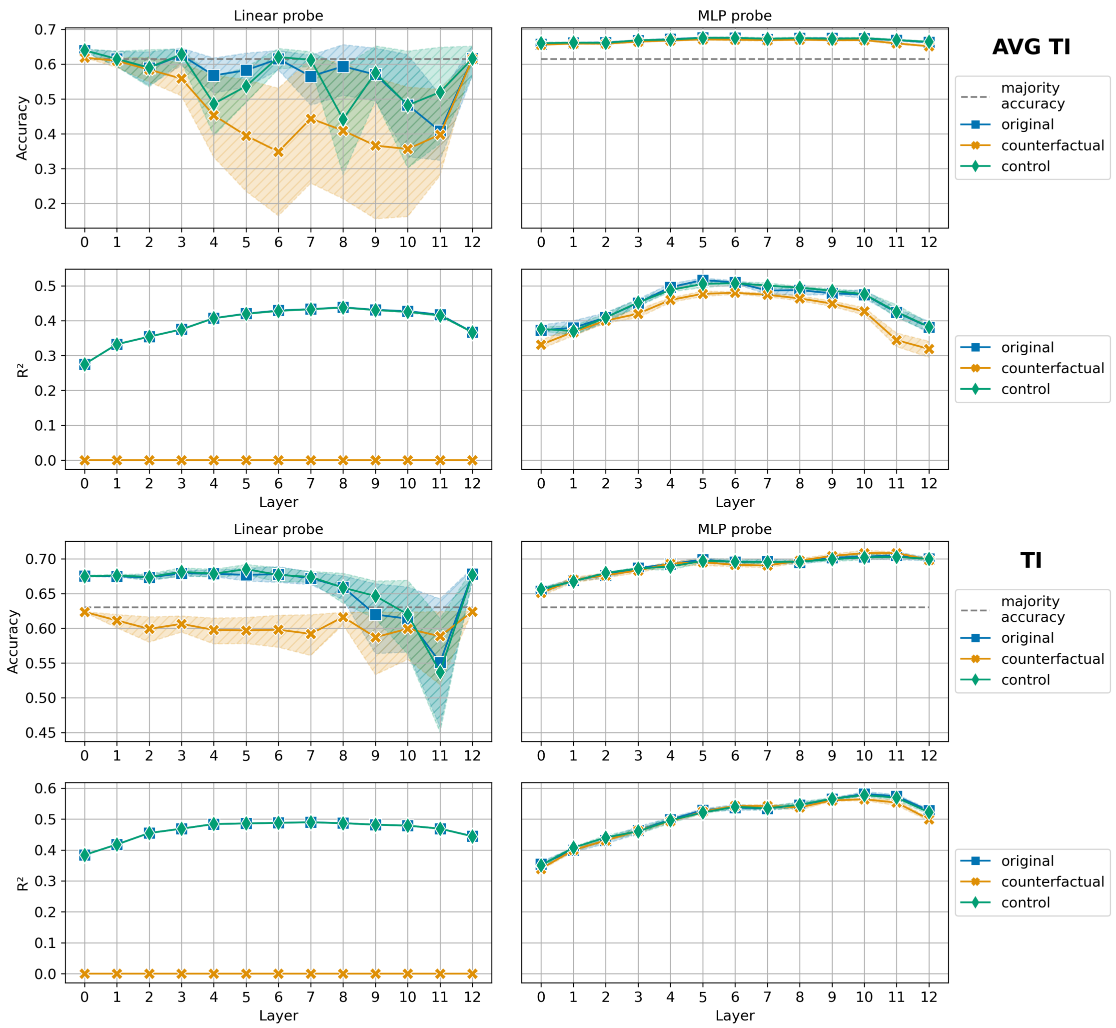

Figure 5.3.: Results of sanity checks for tasks AVG TI & TI. The top two rows depict AVG TI and the bottom two TI. Each individual top row: Binned version (classification). Each individual bottom row: Regression version.

Term Importance (TI)

In contrast to the previous property, the sanity checks for TI seem reasonable to conduct further experiments. In the two bottom rows of Figure 5.3, we can clearly see that the linear model performs above majority accuracy, with the exception of the higher layers 10 and 11. This could again be due to the reasons already mentioned for SEM. The counterfactual intervention consistently impairs the probe's ability to solve the task.

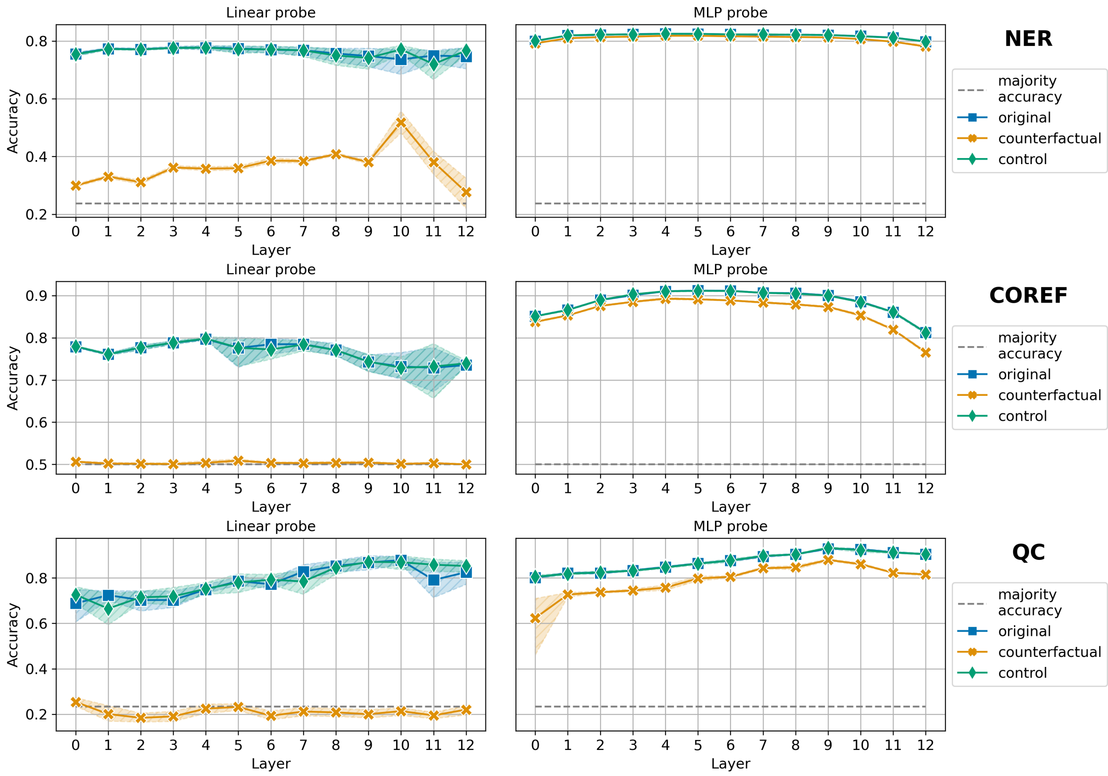

Figure 5.4.: Results of sanity checks for tasks NER, COREF & QC. The rank of the eliminated subspace is chosen as $k = 8$ for NER and $k = 4$ for QC.

Named-Entity Recognition (NER)

For this task, we initially observed that choosing the hyperparameter k of R-LACE as $k = 1$ did not suffice to remove the concept. We already explained why we choose $k = 8$ for this task in the results for the feasibility study from Section 5.3. While the linear model is consistently able to predict the property to a high degree, the intervention clearly damages the probe's ability to categorize the named entities. Still, it performs better than the majority accuracy (see Figure 5.4). Nevertheless, we consider the sanity checks for this task as passed.

Coreference Resolution (COREF)

As observable in the plots for COREF in Figure 5.4 the linear probe is able to predict the property consistently above an accuracy of about 0.75, while the majority accuracy is 0.5. As expected, the non-linear probe outperforms the linear one, and by a significant amount. As we designed COREF to be a binary classification task, the elimination of a rank 1 subspace suffices to push the performance of the linear probe to the majority accuracy. The sanity checks are considered passed for this task.

Question Classification (QC)

The last investigated task QC is also depicted in Figure 5.4. As in the task of NER we had to increase k, in order for R-LACE to successfully remove the linearly present information. A good choice for this task seemed to be $k = 4$. From looking at the left plot for QC of Figure 5.4 it becomes evident that all the criteria for passing the sanity checks are met for this task.

General Findings Across Tasks

We can confirm that in the regression tasks the closed form solution of LACE (see Section 4.1) completely removes the concept of the property ($R^2 = 0$, see Section 4.6.1), visible in the plots on the left-hand side that depict the regression setting of Figures 5.2 and 5.3.

However, when looking at the results for the non-linear probe, we generally see that the influence of the counterfactual intervention is mostly rather modest (BM25, SEM, AVG TI, COREF, QC) if not barely perceptible (TI, NER). This indicates that encoded information about the properties is not solely of linear nature except for the information in the last layer, as there is no further non-linearity applied. In addition, Elazar et al. (2021) show that subsequent layers can recover a property after it is linearly removed. This calls into question whether reliance on linear removal techniques suffices to causally explain non-linear model behavior. Despite this discrepancy, Elazar et al. (2021) follow this approach, as we will, but with this limitation in mind.

Regarding the control representations, one can conclude that the performance is mostly similar to the original representations, which strengthens our confidence in the effectiveness and selectivity of LACE.

In the majority of plots, especially those depicting the MLP probes, we see a decrease in performance at the last layers. Some linear probes are an exception to this having a performance gain at the last layer. It could be attributed to the model's need to store linear information, as there is no further non-linear activation function after that layer. Nevertheless, the initial observation could be due to the known circumstance (Merchant et al., 2020; Singh et al., 2020) that fine-tuning mostly affects the upper layers and lets them forget a proportion of linguistic knowledge they acquired during pre-training. Nonetheless, this effect emerges to different degrees across the tasks, which indicates that some tasks might be more important for retrieval than others. For example, the MLP probe accuracy on the task COREF degrades from 0.9 at layer 9 to 0.81 at layer 12, while for QC, it degrades in the same range from 0.92 to 0.9 (Figure 5.4).

6. Causal Probing Results

In this section, we will present and discuss the results of our main experiment described in Section 4.3 with the properties related to retrieval described in Section 4.5. As the evaluation metric, we use NDCG@10 (Section 4.6.2) as it is the main metric in the TREC 2019 Deep Learning Track and a popular choice for retrieval in the context of web search. We will report absolute values.

Figure 6.1 shows an overview of all the investigated properties. The results for AVG TI can be found in the appendix. The baseline performance is solely depicted at layer 12 since it is only a single value (NDCG@10 of 0.715).

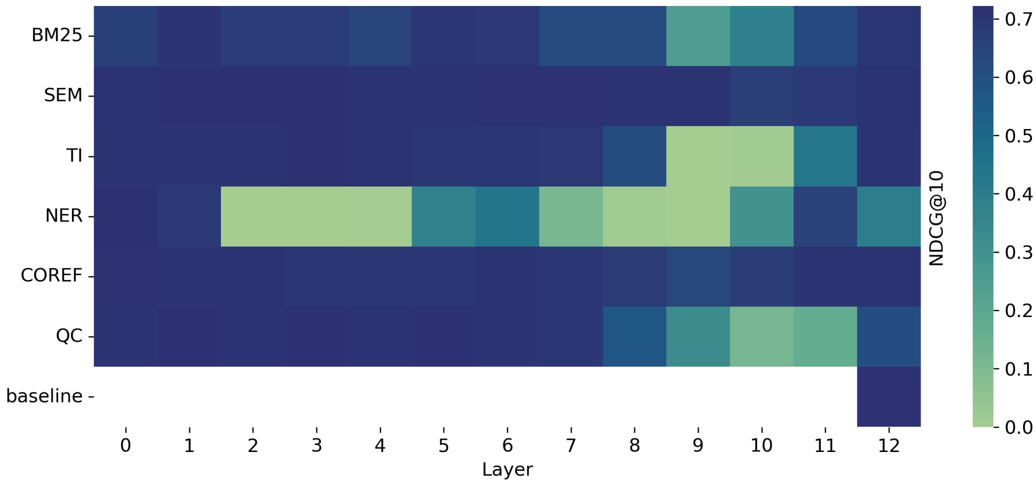

Figure 6.1.: Retrieval performance on the 43 TREC test queries shown per task and intervention layer. The rank of the eliminated subspace is chosen as $k = 8$ for NER, $k = 4$ for QC, and $k = 1$ for all other tasks.

We will now discuss the results for each task and report NDCG@10 values in parenthesis after mentioning a layer.

BM25

In Figure 6.2, we see a distinct performance drop in layers 9 (0.25) and 10 (0.38), but also a moderate decrease across most of the other layers. Referring to the sanity checks

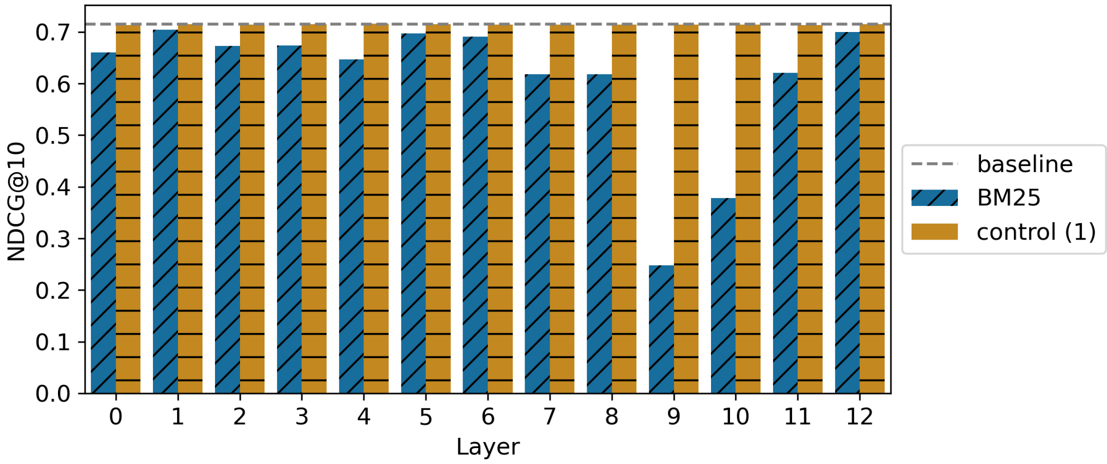

Figure 6.2.: Causal probing results for BM25. We chose a bar plot, as the layer on the x-axis only indicates when the intervention in the forward pass happened. Also, we report the performance of the model on the control representations with a randomly eliminated subspace of the same rank as in the counterfactual intervention.

feasibility study (Section 5.3), we said that the evidence that the property can be removed is incomplete. However, we see that the intervention has the just described noticeable effect, suggesting that our subject model $\mathcal{M}$ relies on some sort of exact term matching. This result partly coincides with the findings of MacAvaney et al. (2022), who showed that TF of query terms remain important for BERT-based ranking models. While TF is not solely determining BM25, it is still a part of it. To pinpoint that BM25 information is most important around the layers 9 and 10 needs further analysis regarding the results of our preliminary experiment from Figure 5.2.

Semantic Similarity (SEM)

As for BM25, we considered the sanity checks for this task incomplete, but we could on average observe a considerable performance degradation caused by the counterfactual intervention. However, when looking at Figure 6.3, we observe almost no influence on the retrieval ability for this task. Only in layers 10 (0.67) and 11 (0.68), we observe a slight performance decrease. This suggests that our subject model $\mathcal{M}$ does not rely on the semantic similarity of a query and a document, at least as we have designed it. Recall that we calculate the similarity by using *GloVe* embeddings, which are context-independent (Section 4.5.2). It sounds plausible that our model does not rely on this simplistic version of semantic similarity and instead relies on encoding the semantics

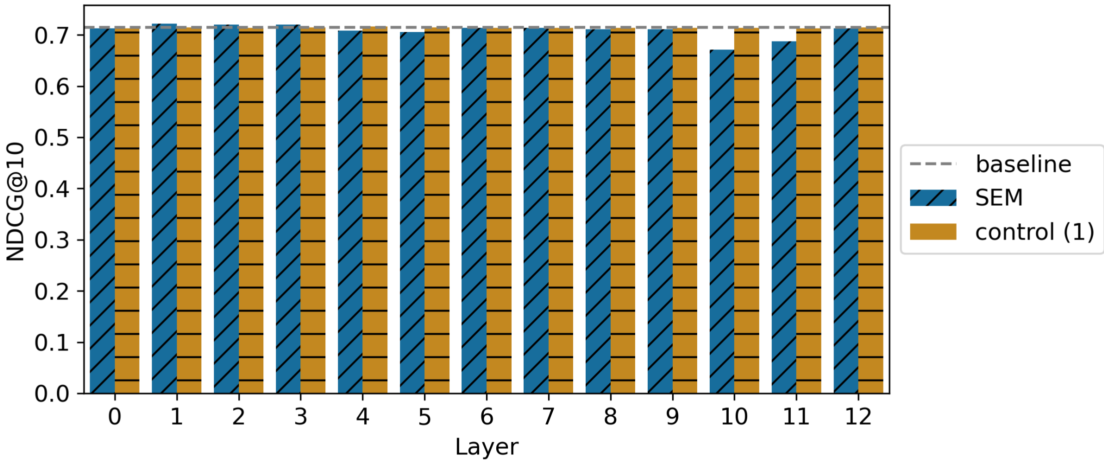

Figure 6.3.: Causal probing results for SEM.

of words solely context-dependent. Therefore, a reasonable approach for future work is to compute the semantic similarity between a query and a document by relying on a contextualized sequence encoder such as the one proposed by Reimers and Gurevych (2019).

Term Importance (TI)

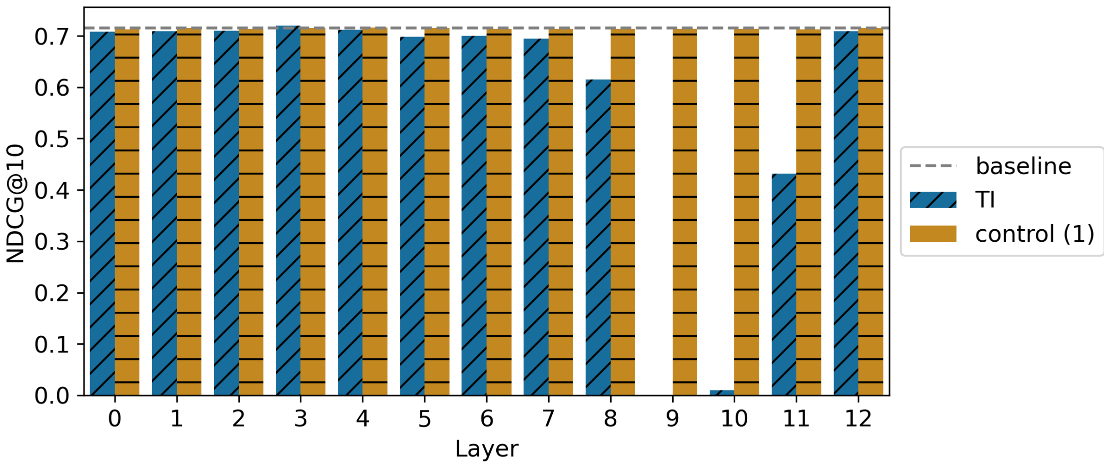

Figure 6.4.: Causal probing results for TI.

For this task (Figure 6.4), we observe a drastic performance decrease with an intervention at layer 9 (0) and 10 (0.01) and a moderate decrease at 8 (0.62) and 11 (0.43). This suggests that at layers 9 and 10, the subject model $\mathcal{M}$ highly relies on assigning importance weights to terms depending on the context and is not able to recover from the linear removal of the property in the subsequent layers. Considering the findings of Formal et al. (2021), one can come to this assumption: Since ColBERT relies on exact matching for important terms to determine relevance, TCT-ColBERT could follow suit. The removal of term importance information might critically impair how the model solves the retrieval and lead to the drastic performance drop we observed. Apart from the fact that Formal et al. (2021) investigated a different, albeit related, model, their analysis regards the re-ranking setting as opposed to our single-stage retrieval. Nevertheless, the assumption might hold.

As our sanity check experiment (Section 5.3) showed that our linear intervention did not affect a non-linear model in solving the task, we can conclude that our subject model $\mathcal{M}$ uses linear information to determine term importance. It would be interesting to investigate whether the model can adapt to the loss of linearly present information and resort to using information that might be non-linearly present. Future work could re-train layers after the counterfactual intervention has been made while remaining the parameters of other layers fixed.

Moreover, these results provide an argument that, in the case of BM25, the linear removal hurts the subject model $\mathcal{M}$ the most at layers 9 and 10 because, as mentioned in Section 4.5.3, TI and BM25 are conceptually similar tasks.

Named-Entity Recognition (NER)

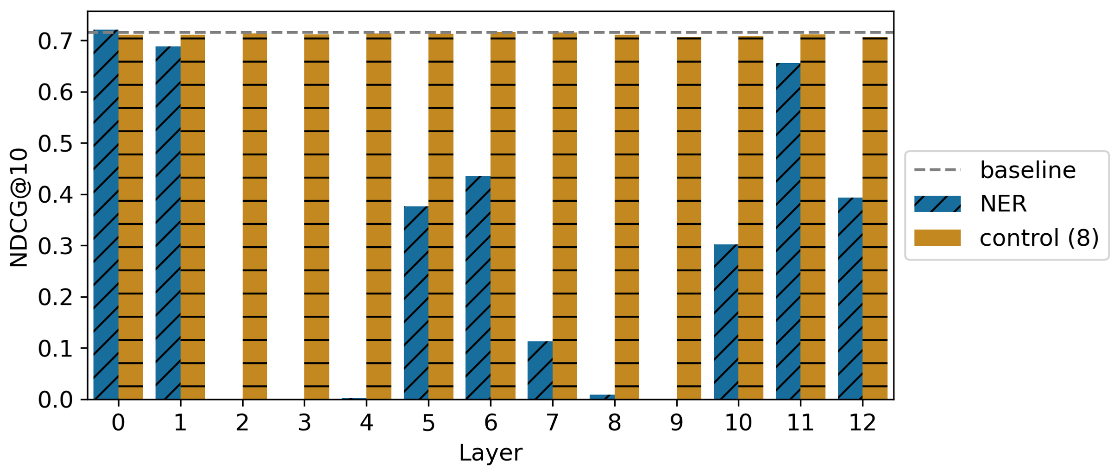

Figure 6.5.: Causal probing results for NER.

As visible in Figure 6.5, the effect of an intervention shows early in layers 2 (0), 3 (0) and 4 (0.003). In the following layers 5 (0.38) and 6 (0.44), the decrease is not as large but still noteworthy. After that, we again see a drastic decrease in layers 7 (0.11), 8 (0.01), and 9 (0.001). Compared to the other tasks, we see the highest decrease at the last layer (0.39) in this task. The results suggest that the model relies on the linear classification of entities throughout most of its layers: Beginning in layers 2-4, somewhat less pronounced in layers 5-6, again stronger in layers 7-9, and worth mentioning also in the last layer. In layer 11 (0.66), the property does not appear to be that significant or could be used in a predominantly non-linear fashion.

As for the previous task, it would be an interesting extension to re-train the model after the intervention happened. We also want to mention that the sharp drop in retrieval performance might be related to the rank of the subspace we eliminate from the representations with the counterfactual intervention. We saw in the previous task that the elimination of a rank-1 subspace is capable of dropping the performance to 0 NDCG@10. Still, it could be that we alter the representations to such a high degree ($k = 8$) that the model struggles to behave in the way it learned to.

Coreference Resolution (COREF)

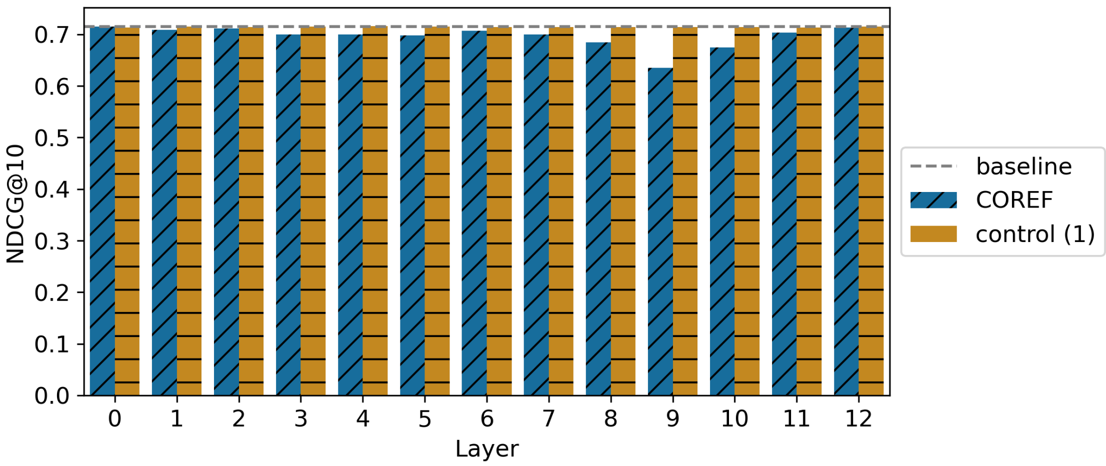

Figure 6.6.: Causal probing results for COREF.

We can observe a very mild decrease in retrieval performance by interventions for this task (Figure 6.6) with a peak in layer 9 (0.635). Wallat et al. (2023) found that models fine-tuned on the task of re-ranking become better in the task of COREF compared to their pre-trained counterpart BERT, while they perform slightly worse in NER. They conclude that COREF might be more important for the task of re-ranking than NER.

For full-stage retrieval with a bi-encoder, we see quite a different picture. COREF seems to not be of much importance, while NER seems vital for our subject model. We hypothesize that since a bi-encoder cannot rely on detecting coreferences spanning from a query to a document via cross-attention, recognizing and possibly linking entities (Section 4.5.7) from a sequence (query or document) and encoding that semantic information takes the place of COREF.

Question Classification (QC)

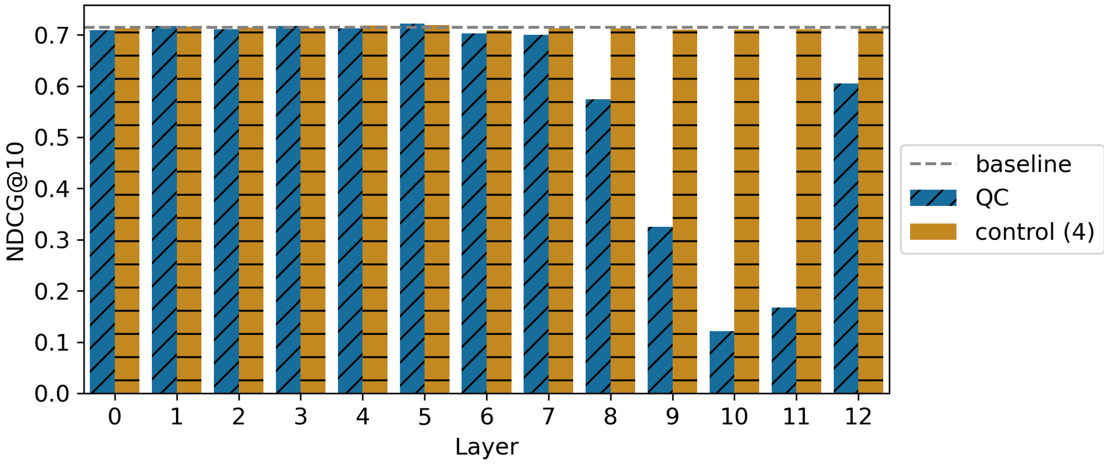

Figure 6.7.: Causal probing results for QC.

In Figure 6.7, we see that our interventions on the query representations have an influence from layers 8 to 12, with a peak in layer 10 (0.12). This suggests that in the upper layers, our subject model $\mathcal{M}$ relies on a linear encoding of the type of information need a query conveys to find relevant documents.

Control Representations

Regarding the control representations for each task shown in the bar plots, we see no influence for a 1-dimensional random subspace eliminated and almost no influence for the 4- and 8-dimensional. This is likely due to the high dimensionality of the BERT hidden representation space and is in line with investigations by Rozanova et al. (2023).

6.1. Discussion

Some of the discussion specific to individual properties has already been provided in the respective sections. Here we want to take a step back and discuss the results at a higher level. Wallat et al. (2023) investigate how ranking properties are distributed in the layers of a re-ranking model. They find that most of the properties are best decoded at intermediate layers. Our finding is that a counterfactual intervention hurts the retrieval performance more in later layers. We argue that this is not a sign of a discrepancy, as it could be that the model might recover the properties after an intervention at early or intermediate layers, like Elazar et al. (2021) observe. Also, the layer that best encodes a property might not be the layer at which the model uses the information.

6.2. Limitations

One limitation of our work regards the circumstance that we can only eliminate an approximation of a property. Ultimately, this poses a threat to the goal of deriving causal explanations. The following could help mitigate this problem: The task datasets need to be sufficiently large, diverse, and preferably error-free. In addition, they have to be constructed with care and thought to be representative of the concept of a property.

There exist other issues that are harder to improve on: We only remove linear information about properties. While we observe that the model certainly relies on linear encoded information, the possibility of important non-linearly encoded information remains. While there are attempts to develop a robust and selective non-linear removal method (Ravfogel et al., 2022b), it remains an open problem.

Also, there might exist spurious correlations with the investigated property in the representations. Our removal technique is unable to disambiguate the information into *spurios* or *relevent*, which diminishes the causal explanations we would like to draw from the removal of a particular property.

7. Conclusion

In this work, we have investigated how to apply the probing paradigm to bi-encoders in the context of retrieval. Moreover, we explored how causal explanations of model behavior can be derived by pursuing a counterfactual white-box approach: We leveraged a debiasing technique (Ravfogel et al., 2022a) to linearly remove the concept of properties that might be related to retrieval, to observe whether their removal negatively impacts our subject model's retrieval ability. For this purpose, we constructed datasets that were supposed to convey the concept of a property.

We can give the following answers to our initial research questions:

RQ 1 Can we confirm the feasibility of *causally probing* our bi-encoder subject model in the context of IR?

RQ 2 On which properties does our bi-encoder rely upon to solve the task of text retrieval?

RQ 3 At which layers are important properties encoded?

To answer **RQ 1**, we conducted two feasibility studies. One served as a sanity check, to validate whether we can conventionally probe a bi-encoder with our constructed task datasets. We could confirm this in the majority of cases but also saw limitations for some of them. The other tested the influence of a hyperparameter choice of the removal technique, which turned out to be crucial for two of the properties we investigated.

After establishing the feasibility of *causally probing* bi-encoders, we investigate **RQ 2**. We detected a rough importance hierarchy of properties for our studied bi-encoder. The matching of multiple mentions of an entity (COREF) and a context-independent semantic matching (SEM) seem to not be important for the model. Moving up the importance hierarchy we can name exact term matching (BM25) and identifying the query's information need (QC), which seem somewhat important for our model. Most prominently our model seems to rely on identifying important terms (TI) and recognizing entities (NER) to determine the relevance of passages.

Concerning **RQ 3**, we found that, except for NER, the properties seem to only be relevant at later layers.

7.1. Future Work

Considering our mentioned limitations and open questions, we propose the following possibilities for future work:

- Test additional properties. We already mentioned a few in Section 4.5.7. Some of them might be more difficult to condense into a dataset to apply our method than others. They could provide further interesting insights into how bi-encoders operate.
- Retraining the model after the counterfactual intervention could give insights into whether the model is able to recover the forgotten information. One option could be to leverage non-linear information that might still exist in the representations after the intervention.
- Improve the quality of task datasets. A key ingredient of our method is a carefully designed task dataset related to a property. Therefore, they should be error-free and conceptually clear and concise. One possible improvement for the property SEM would be to use a sentence encoder to contextually embed queries and documents (see. Section 4.5.2).
- Extend the analysis on different bi-encoder architectures or training regimes. This could provide insights if architectures and training procedures play a role in deciding which properties the model relies on to determine relevance. It could spark new ideas of combining architectures or training procedures, which have been shown to rely on reasonable properties.
- When robust and reliable non-linear removal techniques become available, it would be interesting to repeat our analysis with them. This could give insights into whether models rely on encoding information about certain properties non-linearly.

This thesis covered only a narrow area of understanding the inner workings of LLMs in the context of IR. Still, we hope to have stressed the need to consider causality when interpreting DNNs.

Plagiarism Statement

I hereby confirm that this thesis is my own work and that I have documented all sources used. I have not submitted this thesis for another class or module (or any other means to obtain credit) before.

Hannover, 25.05.2023

(Hauke Tristan Hinrichs)

List of Figures

List of Tables

Bibliography

A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018. doi: 10.1109/ACCESS.2018.2870052. URL `https://doi.org/10.1109/ACCESS.2018.2870052`.

Y. Adi, E. Kermany, Y. Belinkov, O. Lavi, and Y. Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=BJh6Ztuxl`.

A. Anand, L. Lyu, M. Idahl, Y. Wang, J. Wallat, and Z. Zhang. Explainable information retrieval: A survey. *CoRR*, abs/2211.02405, 2022. doi: 10.48550/arXiv.2211.02405. URL `https://doi.org/10.48550/arXiv.2211.02405`.

H. K. Azad and A. Deepak. Query expansion techniques for information retrieval: A survey. *Inf. Process. Manag.*, 56(5):1698–1735, 2019. doi: 10.1016/j.ipm.2019.05.009. URL `https://doi.org/10.1016/j.ipm.2019.05.009`.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.0473`.

L. Bottou. Stochastic gradient descent tricks. In G. Montavon, G. B. Orr, and K. Müller, editors, *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pages 421–436. Springer, 2012. doi: 10.1007/978-3-642-35289-8_25. URL `https://doi.org/10.1007/978-3-642-35289-8_25`.

S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2014. ISBN 978-0-521-83378-3. doi: 10.1017/CBO9780511804441. URL `https://web.stanford.edu/%7Eboyd/cvxbook/`.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse,

M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html`.

A. Câmara and C. Hauff. Diagnosing BERT with retrieval heuristics. In J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins, editors, *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part I*, volume 12035 of *Lecture Notes in Computer Science*, pages 605–618. Springer, 2020. doi: 10.1007/978-3-030-45439-5_40. URL `https://doi.org/10.1007/978-3-030-45439-5_40`.

R. Chen, L. Gallagher, R. Blanco, and J. S. Culpepper. Efficient cost-aware cascade ranking in multi-stage retrieval. In N. Kando, T. Sakai, H. Joho, H. Li, A. P. de Vries, and R. W. White, editors, *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 445–454. ACM, 2017. doi: 10.1145/3077136.3080819. URL `https://doi.org/10.1145/3077136.3080819`.

J. Cheng, L. Dong, and M. Lapata. Long short-term memory-networks for machine reading. In J. Su, X. Carreras, and K. Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 551–561. The Association for Computational Linguistics, 2016. doi: 10.18653/v1/d16-1053. URL `https://doi.org/10.18653/v1/d16-1053`.

J. Choi, E. Jung, S. Lim, and W. Rhee. Finding inverse document frequency information in BERT. *CoRR*, abs/2202.12191, 2022. URL `https://arxiv.org/abs/2202.12191`.

A. Conneau, G. Kruszewski, G. Lample, L. Barrault, and M. Baroni. What you can cram into a single \$&!#* vector: Probing sentence embeddings for linguistic properties. In I. Gurevych and Y. Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2126–2136. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1198. URL `https://aclanthology.org/P18-1198/`.

N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and E. M. Voorhees. Overview of the TREC 2019 deep learning track. *CoRR*, abs/2003.07820, 2020. URL `https://arxiv.org/abs/2003.07820`.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL `https://doi.org/10.18653/v1/n19-1423`.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

Y. Elazar, S. Ravfogel, A. Jacovi, and Y. Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Trans. Assoc. Comput. Linguistics*, 9:160–175, 2021. doi: 10.1162/tacl_a_00359. URL `https://doi.org/10.1162/tacl_a_00359`.

K. Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and GPT-2 embeddings. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 55–65. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1006. URL `https://doi.org/10.18653/v1/D19-1006`.

A. Feder, N. Oved, U. Shalit, and R. Reichart. Causalm: Causal model explanation through counterfactual language models. *Comput. Linguistics*, 47(2):333–386, 2021. doi: 10.1162/coli_a_00404. URL `https://doi.org/10.1162/coli_a_00404`.

T. Formal, B. Piwowarski, and S. Clinchant. A white box analysis of colbert. In D. Hiemstra, M. Moens, J. Mothe, R. Perego, M. Potthast, and F. Sebastiani, editors, *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II*, volume 12657 of *Lecture Notes in Computer Science*, pages 257–263. Springer, 2021. doi: 10.1007/978-3-030-72240-1_23. URL `https://doi.org/10.1007/978-3-030-72240-1_23`.

T. Formal, B. Piwowarski, and S. Clinchant. Match your words! A study of lexical matching in neural information retrieval. In M. Hagen, S. Verberne, C. Macdonald, C. Seifert, K. Balog, K. Nørvåg, and V. Setty, editors, *Advances in Information Retrieval - 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10-14, 2022, Proceedings, Part II*, volume 13186 of *Lecture Notes in Computer*

Science, pages 120–127. Springer, 2022. doi: 10.1007/978-3-030-99739-7_14. URL https://doi.org/10.1007/978-3-030-99739-7_14.

G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Commun. ACM*, 30(11):964–971, nov 1987. ISSN 0001-0782. doi: 10.1145/32206.32212. URL https://doi.org/10.1145/32206.32212.

N. Ganguly, D. Fazlija, M. Badar, M. Fisichella, S. Sikdar, J. Schrader, J. Wallat, K. Rudra, M. Koubarakis, G. K. Patro, W. Z. E. Amri, and W. Nejdl. A review of the role of causality in developing trustworthy AI systems. *CoRR*, abs/2302.06975, 2023. doi: 10.48550/arXiv.2302.06975. URL https://doi.org/10.48550/arXiv.2302.06975.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

J. Guo, Y. Cai, Y. Fan, F. Sun, R. Zhang, and X. Cheng. Semantic models for the first-stage retrieval: A comprehensive review. *ACM Trans. Inf. Syst.*, 40(4):66:1–66:42, 2022. doi: 10.1145/3486250. URL https://doi.org/10.1145/3486250.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

J. Hewitt and P. Liang. Designing and interpreting probes with control tasks. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2733–2743. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1275. URL https://doi.org/10.18653/v1/D19-1275.

D. E. Hilt and D. W. Seegrist. Ridge: a computer program for calculating ridge regression estimates. 1977.

G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL http://arxiv.org/abs/1503.02531.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9: 1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

M. Honnibal, I. Montani, S. V. Landeghem, and A. Boyd. spacy: Industrial-strength natural language processing in python. 2020. doi: 10.5281/zenodo.1212303.

J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572. URL `https://doi.org/10.1109/TBDATA.2019.2921572`.

J. D. Kelleher, B. MacNamee, and A. D'Arcy. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies.* MIT Press, Cambridge, MA, 2015. ISBN 978-0-262-02944-5.

O. Khattab and M. Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In J. X. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. Wen, and Y. Liu, editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 39–48. ACM, 2020. doi: 10.1145/3397271.3401075. URL `https://doi.org/10.1145/3397271.3401075`.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

N. Kolitsas, O. Ganea, and T. Hofmann. End-to-end neural entity linking. In A. Korhonen and I. Titov, editors, *Proceedings of the 22nd Conference on Computational Natural Language Learning, CoNLL 2018, Brussels, Belgium, October 31 - November 1, 2018*, pages 519–529. Association for Computational Linguistics, 2018. doi: 10.18653/v1/k18-1050. URL `https://doi.org/10.18653/v1/k18-1050`.

K. Lasri, T. Pimentel, A. Lenci, T. Poibeau, and R. Cotterell. Probing for the usage of grammatical number. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8818–8831. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.603. URL `https://doi.org/10.18653/v1/2022.acl-long.603`.

T. Lau and E. Horvitz. Patterns of search: Analyzing and modeling web query refinement. In J. Kay, editor, *UM99 User Modeling*, pages 119–128, Vienna, 1999. Springer Vienna. ISBN 978-3-7091-2490-1.

K. Lee, L. He, M. Lewis, and L. Zettlemoyer. End-to-end neural coreference resolution. In M. Palmer, R. Hwa, and S. Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 188–197. Association for Computational Linguistics, 2017. doi: 10.18653/v1/d17-1018. URL `https://doi.org/10.18653/v1/d17-1018`.

X. Li and D. Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. URL `https://aclanthology.org/C02-1150`.

J. Lin, R. F. Nogueira, and A. Yates. *Pretrained Transformers for Text Ranking: BERT and Beyond*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2021a. doi: 10.2200/S01123ED1V01Y202108HLT053. URL `https://doi.org/10.2200/S01123ED1V01Y202108HLT053`.

S. Lin, J. Yang, and J. Lin. Distilling dense representations for ranking using tightly-coupled teachers. *CoRR*, abs/2010.11386, 2020. URL `https://arxiv.org/abs/2010.11386`.

S. Lin, J. Yang, and J. Lin. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In A. Rogers, I. Calixto, I. Vulic, N. Saphra, N. Kassner, O. Camburu, T. Bansal, and V. Shwartz, editors, *Proceedings of the 6th Workshop on Representation Learning for NLP, RepL4NLP@ACL-IJCNLP 2021, Online, August 6, 2021*, pages 163–173. Association for Computational Linguistics, 2021b. doi: 10.18653/v1/2021.repl4nlp-1.17. URL `https://doi.org/10.18653/v1/2021.repl4nlp-1.17`.

T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, and Y. Marton, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421. The Association for Computational Linguistics, 2015. doi: 10.18653/v1/d15-1166. URL `https://doi.org/10.18653/v1/d15-1166`.

S. MacAvaney, S. Feldman, N. Goharian, D. Downey, and A. Cohan. ABNIRML: analyzing the behavior of neural IR models. *Trans. Assoc. Comput. Linguistics*, 10:224–239, 2022. doi: 10.1162/tacl_a_00457. URL `https://doi.org/10.1162/tacl_a_00457`.

C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008. ISBN 978-0-521-86571-5. URL `http://nlp.stanford.edu/IR-book/information-retrieval-book.html`.

A. Merchant, E. Rahimtoroghi, E. Pavlick, and I. Tenney. What happens to BERT embeddings during fine-tuning? In A. Alishahi, Y. Belinkov, G. Chrupala, D. Hupkes, Y. Pinter, and H. Sajjad, editors, *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2020, Online, November 2020*, pages 33–44. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.blackboxnlp-1.4. URL `https://doi.org/10.18653/v1/2020.blackboxnlp-1.4`.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In Y. Bengio and Y. LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL `http://arxiv.org/abs/1301.3781`.

D. Neimark, O. Bar, M. Zohar, and D. Asselmann. Video transformer network. In *IEEE/CVF International Conference on Computer Vision Workshops, ICCVW 2021, Montreal, BC, Canada, October 11-17, 2021*, pages 3156–3165. IEEE, 2021. doi: 10.1109/ICCVW54120.2021.00355. URL `https://doi.org/10.1109/ICCVW54120.2021.00355`.

T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. MS MARCO: A human generated machine reading comprehension dataset. In T. R. Besold, A. Bordes, A. S. d'Avila Garcez, and G. Wayne, editors, *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. URL `https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf`.

R. F. Nogueira and K. Cho. Passage re-ranking with BERT. *CoRR*, abs/1901.04085, 2019. URL `http://arxiv.org/abs/1901.04085`.

M. Palmer, D. Gildea, and N. Xue. *Semantic Role Labeling*. Synthesis lectures on human language technologies. Morgan & Claypool Publishers, 2010. ISBN 9781598298314. URL `https://books.google.de/books?id=6C1Ag3NUqNEC`.

J.-S. Pang and M. Razaviyayn. *A unified distributed algorithm for non-cooperative games*, page 101–134. Cambridge University Press, 2016. doi: 10.1017/CBO9781316162750.005.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014. doi: 10.3115/v1/d14-1162. URL `https://doi.org/10.3115/v1/d14-1162`.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In M. A. Walker, H. Ji, and A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics, 2018a. doi: 10.18653/v1/n18-1202. URL `https://doi.org/10.18653/v1/n18-1202`.

M. E. Peters, M. Neumann, L. Zettlemoyer, and W. Yih. Dissecting contextual word embeddings: Architecture and representation. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1499–1509. Association for Computational Linguistics, 2018b. doi: 10.18653/v1/d18-1179. URL `https://doi.org/10.18653/v1/d18-1179`.

T. Pimentel, J. Valvoda, R. H. Maudslay, R. Zmigrod, A. Williams, and R. Cotterell. Information-theoretic probing for linguistic structure. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4609–4622. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.420. URL `https://doi.org/10.18653/v1/2020.acl-main.420`.

Y. Qiao, C. Xiong, Z. Liu, and Z. Liu. Understanding the behaviors of BERT in ranking. *CoRR*, abs/1904.07531, 2019. URL `http://arxiv.org/abs/1904.07531`.

S. Ravfogel, Y. Elazar, H. Gonen, M. Twiton, and Y. Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7237–7256. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.647. URL `https://doi.org/10.18653/v1/2020.acl-main.647`.

S. Ravfogel, M. Twiton, Y. Goldberg, and R. Cotterell. Linear adversarial concept erasure. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 18400–18421. PMLR, 2022a. URL `https://proceedings.mlr.press/v162/ravfogel22a.html`.

S. Ravfogel, F. Vargas, Y. Goldberg, and R. Cotterell. Adversarial concept erasure in kernel space. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 6034–6055. Association for Computational Linguistics, 2022b. URL `https://aclanthology.org/2022.emnlp-main.405`.

A. Ravichander, Y. Belinkov, and E. H. Hovy. Probing the probing paradigm: Does probing accuracy entail task relevance? In P. Merlo, J. Tiedemann, and R. Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 3363–3377. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.eacl-main.295. URL `https://doi.org/10.18653/v1/2021.eacl-main.295`.

N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1410. URL `https://doi.org/10.18653/v1/D19-1410`.

D. Reisinger, R. Rudinger, F. Ferraro, C. Harman, K. Rawlins, and B. Van Durme. Semantic proto-roles. *Transactions of the Association for Computational Linguistics*, 3:475–488, 2015. doi: 10.1162/tacl_a_00152. URL `https://aclanthology.org/Q15-1034`.

D. Rennings, F. Moraes, and C. Hauff. An axiomatic approach to diagnosing neural IR models. In L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, and D. Hiemstra, editors, *Advances in Information Retrieval*, pages 489–503, Cham, 2019. Springer International Publishing. ISBN 978-3-030-15712-8.

S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389, 01 2009. doi: 10.1561/1500000019. URL `https://www.staff.city.ac.uk/~sbrp622/papers/foundations_bm25_review.pdf`.

S. E. Robertson, S. Jones, and K. Spärck. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976. doi: 10.1002/asi.4630270302. URL `http://dx.doi.org/10.1002/asi.4630270302`.

S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *Text Retrieval Conference*, 1994.

J. Rozanova, M. Valentino, L. C. Cordeiro, and A. Freitas. Interventional probing in high dimensions: An NLI case study. In A. Vlachos and I. Augenstein, editors, *Findings of the Association for Computational Linguistics: EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pages 2444–2455. Association for Computational Linguistics, 2023. URL `https://aclanthology.org/2023.findings-eacl.188`.

J. Singh and A. Anand. EXS: explainable search using local model agnostic interpretability. In J. S. Culpepper, A. Moffat, P. N. Bennett, and K. Lerman, editors, *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 770–773. ACM, 2019. doi: 10.1145/3289600.3290620. URL `https://doi.org/10.1145/3289600.3290620`.

J. Singh, J. Wallat, and A. Anand. Bertnesia: Investigating the capture and forgetting of knowledge in BERT. In A. Alishahi, Y. Belinkov, G. Chrupala, D. Hupkes, Y. Pinter, and H. Sajjad, editors, *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2020, Online, November 2020*, pages 174–183. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.blackboxnlp-1.17. URL `https://doi.org/10.18653/v1/2020.blackboxnlp-1.17`.

A. Slobodkin, L. Choshen, and O. Abend. Mediators in determining what processing BERT performs first. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 86–93. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.8. URL `https://doi.org/10.18653/v1/2021.naacl-main.8`.

I. Tenney, D. Das, and E. Pavlick. BERT rediscovers the classical NLP pipeline. In A. Korhonen, D. R. Traum, and L. Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4593–4601. Association for Computational Linguistics, 2019a. doi: 10.18653/v1/p19-1452. URL `https://doi.org/10.18653/v1/p19-1452`.

I. Tenney, P. Xia, B. Chen, A. Wang, A. Poliak, R. T. McCoy, N. Kim, B. V. Durme, S. R. Bowman, D. Das, and E. Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b. URL `https://openreview.net/forum?id=SJzSgnRcKX`.

H. Tuy. Minimax theorems revisited. *Acta Mathematica Vietnamica*, 29:217–229, 01 2004. URL `http://journals.math.ac.vn/acta/pdf/0403217.pdf`.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio,

H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`.

M. Verma and D. Ganguly. LIRME: locally interpretable ranking model explanation. In B. Piwowarski, M. Chevalier, É. Gaussier, Y. Maarek, J. Nie, and F. Scholer, editors, *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 1281–1284. ACM, 2019. doi: 10.1145/3331184.3331377. URL `https://doi.org/10.1145/3331184.3331377`.

P. Verma and J. Berger. Audio transformers: Transformer architectures for large scale audio understanding. adieu convolutions. *CoRR*, abs/2105.00335, 2021. URL `https://arxiv.org/abs/2105.00335`.

E. Voita and I. Titov. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.14. URL `https://aclanthology.org/2020.emnlp-main.14`.

M. Völske, A. Bondarenko, M. Fröbe, B. Stein, J. Singh, M. Hagen, and A. Anand. Towards axiomatic explanations for neural ranking models. In F. Hasibi, Y. Fang, and A. Aizawa, editors, *ICTIR '21: The 2021 ACM SIGIR International Conference on the Theory of Information Retrieval, Virtual Event, Canada, July 11, 2021*, pages 13–22. ACM, 2021. doi: 10.1145/3471158.3472256. URL `https://doi.org/10.1145/3471158.3472256`.

J. Wallat, F. Beringer, A. Anand, and A. Anand. Probing bert for ranking abilities. In J. Kamps, L. Goeuriot, F. Crestani, M. Maistro, H. Joho, B. Davis, C. Gurrin, U. Kruschwitz, and A. Caputo, editors, *Advances in Information Retrieval*, pages 255–273, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-28238-6.

Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in time series: A survey, 2022. URL `https://arxiv.org/abs/2202.07125`.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL `http://arxiv.org/abs/1609.08144`.

A. Appendix

Tokenizer Issues

For the token-level tasks we use the *spaCy* tokenizer to compute the spans of tokens in passages. However, when we want to know which hidden representations of tokens correspond to these spans we have to account for the differences of the *WordPiece* (Wu et al., 2016) BERT tokenizer and the *spaCy* tokenizer. We achieve this by re-tokenizing the passage previously tokenized with the *spaCy* tokenizer with the BERT tokenizer, eliminating all problems arising from the discrepancy that the *spaCy* tokenizer does not employ *WordPiece* tokenization. However, the re-tokenization introduces problems with the tokenization of contractions, which we solve by adding exceptions to the *spaCy* tokenizer. Lastly, there are solvable issues around special characters like '°' in '°C' or '£'. Still, some problems evolving around special characters remained, which we decided to solve by manually removing those from the datasets in order to clean them. The number of removed samples per task can be found in Table A.1.

Task	**# manually removed samples**
TI	5
NER	3
COREF	1

Table A.1.: Number of manually removed samples from token-level task datasets due to tokenizer issues.

Additional Overview Heatmap Recall@1000

Since we only report NDCG@10 it could be that a value of 0 might be misleading, i.e., the first 10 documents might be non-relevant, but after the first 10 the model retrieved relevant documents. Therefore, we include an overview reporting Recall@1000 in Figure A.1. The distribution roughly resembles the one from Figure 6.1.

AVG TI

Since neither our sanity checks could be considered passed nor we observe an influence on the retrieval performance for this task, we come to the same conclusion as before

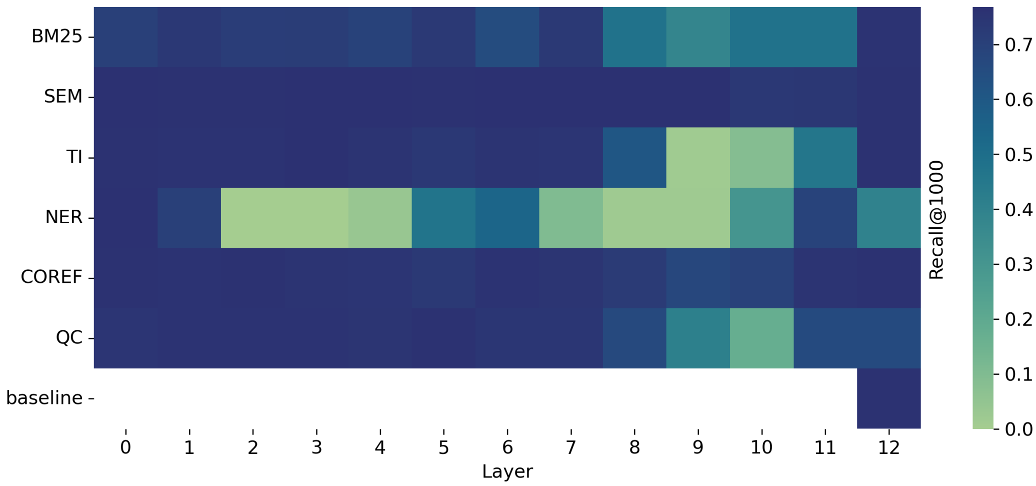

Figure A.1.: Causal probing result overview. The layer on the x-axis depicts when the intervention happened. We report Recall@1000.

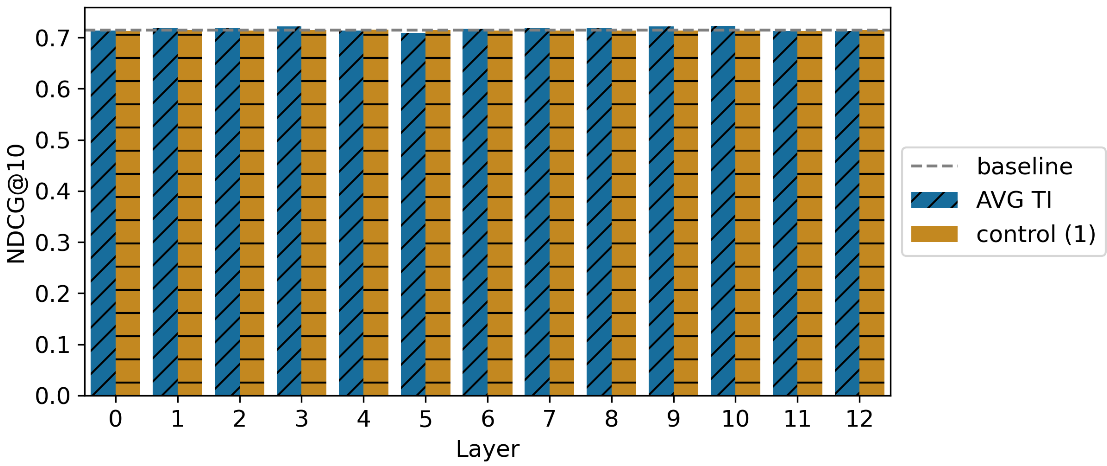

Figure A.2.: Causal probing result for AVG TI.

(Section 5.3): The design of this task might not be descriptive enough for the probe to understand the concept of this property.