Classe di Scienze
Corso di perfezionamento in
Data Science
XXXV ciclo

Transfomer Models: From Model Inspection to Applications in Patents

INF/01 - Informatica

Candidato
dr. Giovanni PUCCETTI

Relatore
Professore Gualtiero Fantoni
Ricercator Felice Dell'Orletta
Supervisore
Fosca GIANNOTTI

Abstract

Natural Language Processing is used to address several tasks, linguistic related ones, e.g. part of speech tagging (Màrquez and Rodríguez, 1998), dependency parsing (Nivre et al., 2016), and downstream tasks, e.g. machine translation, sentiment analysis. To tackle these tasks, dedicated approaches have been developed over time.

A methodology that makes it possible to increase performance on all of them in a unified manner is language modeling (Bengio et al., 2003), this is done by pre-training a model to replace masked tokens in large amounts of text, either randomly within chunks of text or sequentially one after the other, to develop general purpose representations that can be used to improve performance in many downstream tasks at once.

The neural network architecture currently best performing this task is the transformer (Vaswani et al., 2017), a model based on attention, a technique that allows to focus on specific parts of a sequence. Together with this architecture, model size and data scale have been found to be essential to the development of information-rich representations (Liu et al., 2019b; Raffel et al., 2020). The availability of large scale datasets and the use of models with billions of parameters is currently identified as the most effective path towards better representations of text.

However, with large models, comes the difficulty in interpreting the output they provide, at the same time, the use of datasets with billions of tokens makes it ever more difficult to identify confounding factors between training and validation data. Therefore, several studies have been carried out to investigate the representations provided by transformers models trained on large scale datasets (Conneau et al., 2018; Rogers et al., 2020).

In this thesis I investigate these models from several perspectives, I study the linguistic properties of the representations provided by BERT (Devlin et al., 2019), a language model mostly trained on the English Wikipedia, to understand if the information it codifies is localized within specific entries of the vector representation (Puccetti et al., 2021b). Doing this I identify special weights that show high relevance to several distinct linguistic probing tasks. Following this I investigate the cause of these special weights, and link them to token distribution and special tokens (Puccetti et al., 2022). Subsequently, I investigate the effect of different pre-training runs on downstream tasks with respect to the number of pre-training steps to understand if such special weights reveal different patterns.

To complement this general purpose analysis and extend it to more specific use cases, given the wide range of applications for language models, I study their effectiveness on technical documentation, specifically, patents.

I use both general purpose and dedicated models, to identify domain-specific entities such as users of the inventions and technologies (Puccetti et al., 2023) or to segment patents text. I always study performance analysis complementing it with careful measurements of data and model properties to understand if the conclusions drawn for general purpose models hold in this context as well.

Contents

Contents

CHAPTER 1

Introduction

This thesis focuses on studying Transformer-Based Language Models, a family of models initially developed to tackle Natural Language Processing (NLP) tasks. It works towards improving the understanding of these models inner workings and their application to the extraction of technical information from patents.

The widespread of Transformer Based Language Models (LM), also called Neural Language Models (NLM) or Large Language Models (LLM), is the result of different concurring factors. The rise of pre-trained language models (Peters et al., 2018) has led to significant improvements in several (if not every) NLP task, this happened in parallel with the development of transformers (Vaswani et al., 2017) and applied at large scale in a pre-train/fine-tune paradigm (Devlin et al., 2019).

The impact of these methodologies has been so strong that benchmark datasets (Wang et al., 2018) are being updated at a fast pace (Wang et al., 2019) to provide more challenging tasks, and need to be increasingly larger (Srivastava et al., 2023). The main drawback of such large scale approaches, despite the improved performances, is the lack of interpretability. The dense high dimensional representations created by large models, do not allow for any insight of the properties they encode.

This work focuses first on understanding these models inner workings from a linguistic and mechanistic perspective and after that on applications of such models in fields where this knowledge is relevant, such as information retrieval from patents.

Since effective transformer models, e.g. BERT (Devlin et al., 2019), are so large, their implicit knowledge is more easily determined a posteriori, by designing tasks that require a specific linguistic skill to be tackled (Linzen and Baroni, 2021) or by investigating to what extent certain information is encoded within contextualized representations, e.g. defining probing classifiers trained to predict a variety of language phenomena (Conneau et al., 2018; Hewitt and Manning, 2019b; Tenney et al., 2019a).

In line with the latter approach and with works aimed at investigating how the information is arranged within neural models representations (Baan et al., 2019; Dalvi et al., 2019; Lakretz et al., 2019), the first part of this work focuses on an in-depth investigation aimed at understanding how the information encoded by BERT is arranged within its representation (Puccetti et al., 2021b).

This reveals a noticeable degree of knowledge localization within the model hidden states and that in this way one can cluster different tasks based on their linguistic nature. In particular, I show that there are few dimensions that are consistently needed to encode the linguistic tasks, regardless of how structured is

the information under probing. The same dimensions have also been found by works focusing on more in depth analysis of transformers (Kovaleva et al., 2021).

Indeed, while a posteriori analyses of the output of transformers are informative and help understand the kind of knowledge that is encoded in the representations provided by these models, to have a better view of the mechanics underlying them, studying their parameters more closely can be valuable too. This is relevant since, Transformer-based language models are heavily overparametrized, which explains the success of pruning methods reducing the model size by up to 30-40% (Gordon et al., 2020; Sanh et al., 2020; Prasanna et al., 2020; Chen et al., 2020, inter alia) without a significant drop in performance. However, it has been shown that multiple Transformer-based language models are highly sensitive to the removal of *outliers* Kovaleva et al. (2021).

These are parameters in the output element of a Transformer layer, the magnitude of which is unusually large within the layer, consistently in the same dimension across the model layers. They contribute to the vector representation of different tokens adding a similar component to all of them, increasing the average similarity and thus making the space less isotropic.

Initially spoken of in Kovaleva et al. (2021), although these parameters constitute less than 0.0001% of the full BERT Devlin et al. (2019) model, removing them significantly degrades BERT's performance both as a LM and after fine-tuning.

Puccetti et al. (2021b) found that the same parameters are particularly relevant in several linguistic probing tasks.

In this work I explore the outlier phenomenon and show that these parameters' magnitude grows during the initial part of pre-training, clarify the role they play on self-attention patterns and connect them with the distribution of tokens in text (Puccetti et al., 2022).

Extending further, Dettmers et al. (2022) investigate this phenomenon on large scale models, up to 176 billions parameters and show that it behaves like an emergent property. That is, scaling up models shows more outliers and they appear abruptly after surpassing given size thresholds, instead of growing continuously with the model architecture.

As mentioned, outliers removal disrupts transformers performance on downstream tasks. Such models are used in a transfer learning framework: first the model is pre-trained on a large volume of unlabeled data using compatible training objective such as masked language modeling. At the following stage the model (possibly with minor task-specific modifications) is fine-tuned, i.e. trained again on a data corresponding to a given task, commonly referred to as a "downstream task".

After showing how these parameters grow during the pre-training of language models and knowing that outliers' removal is harmful for performance on downstream tasks, to further develop the understanding of transformer-based language models, I focus on investigating the relation between pre-training and fine-tuning.

Indeed, while this is the main paradigm, there are reports suggesting that BERT can be successfully fine-tuned on standard Neural Language Understanding (NLU) tasks even starting from randomly initialized models Kovaleva et al. (2019) when pre-trained on non-linguistic tasks Kao and Lee (2021) and even shuffling word order (Sinha et al., 2021). I address the issue of quantifying the respective contribution of pre-training and fine-tuning in the success of LMs, using distinct architecture variants and hard adversarial datasets for evaluation.

Thus far I mention investigating language models from several perspectives, developing an understanding of the amount and localization of implicit linguistic knowledge they store, studying the role of special parameters within the model architecture relating their presence with inherent properties of language and finally I investigate the relation between the two components of training language models on downstream tasks. This improves the understanding of these models and allows a more aware use in downstream tasks.

One of the limitations of the mentioned analyses is that they are based on models trained on general purpose datasets, evaluated in open domain text. To address this I study applications of transformers on patents, a natural test bench for the type of analysis that is missing. Indeed, patents are a vast corpus of openly available documents written in a unique language meant to be technically sound and legally binding.

In particular, I focus on the task of extracting technologies and technical entities from patents. I try and extract these entities while comparing transformer models with methodologies based on different

approaches, most importantly not based on training distributional models, but rather on the efforts of field experts, with the goal of providing new different insights on how transformer models work when adapted to different contexts.

Besides its challenging nature, technology extraction from patents is at the core of a vast number of applications that can help decision makers in predicting core and emerging technologies (Huang et al., 2021), diffusion of technologies (Daim et al., 2006), convergence (Karvonen and Kässi, 2013) and portfolio analysis (Ernst, 2003). Patents are among the best source of information to reach these goals (Joung and Kim, 2017) because they contain rich technical details and they are also among the most complex textual sources to analyse automatically because of the mix of technical and juridical jargon.

A patent provides the technical description of an invention in a sufficiently clear and complete manner to enable a person skilled in the art (i.e. someone having the relevant technical information publicly disclosed at the time of the invention) to replicate the invention without any additional creative activity (Art. 83 EPC) (Lidén and Setréus, 2011). A patent is designed to disclose the minimum content that makes understandable and reproducible the invention. The use of juridical jargon makes the document legally binding, eligible to protect the invention.

Patent texts create a barrier to the access to one of the widest technical open access resources. It is believed that between 70% and 90% of the information about technologies can only be found in patents (Asche, 2017). Patent analysis is a valuable approach for deriving information about an industry or technology for forecasting (Daim et al., 2006), competitive analysis (Thorleuchter et al., 2010), technology trend analysis (Tseng et al., 2011), and for avoiding infringement (Yu and Zhang, 2019). This information may be obtained by the use of either bibliometric analysis and text mining. The former involves the analysis of meta-data, such as citations, assignee, inventors, and International Patent Classification classes (Cho and Kim, 2014). The latter aims to extract relevant information from unstructured text, that includes title, abstract, claims, state-of-the-art description, and other records (Tseng et al., 2007).

As affirmed by Vicente-Gomila et al. (2021), text mining enhances traditional measures based on bibliometric data in forecasting technological change. Recently it has been shown that text mining is more effective than metadata analysis for measuring novelty and impact of a patent. In Arts et al. (2021), the authors use text mining techniques to identify new technologies and measure patent novelty, detecting uni-grams (single word, well-known as keywords), bi-grams and tri-grams (two or three consecutive words, also called keyphrases) in title, abstract, or claims of a patent. They consider as emerging technologies a word or a words combination appearing for the first time in the text, achieving remarkable results with generic keywords and keyphrases. Other studies involve text mining from patent for identifying emerging technologies (Porter et al., 2019; Ranaei et al., 2020; Zhou et al., 2020; Jang et al., 2021; Sarica et al., 2020) or exploring the convergence phenomena among technological fields (Gustafsson et al., 2015; Song et al., 2017). At the state-of-the-art, text mining techniques for technological analysis focus on generic terms and not on specific ones for investigating the technological change. This is a limitation given the quantity and quality of technical information contained in patents, therefore novel approaches are needed to enable these contents.

I use Natural Language Processing (NLP) to recognize the technologies mentioned in a patent. I reach this goal by solving a well known task in NLP called Named Entity Recognition (NER). NER systems are widely used to extract general entities (such as persons' names, cities, dates and times), but there is still a lack of tools able to extract technically relevant information (Fantoni et al., 2013; Chiarello et al., 2018a, 2020).

I focus on measuring how the use of transformer-based language models can help in this context. I test three methods, lexicon-based, rule-based and distributional NER. Lexicon-based and rule-based NER exploit pre-defined lists of entities (lexicons) or rules-driven expert systems. Distributional NER exploits machine learning and in particular language models, eliciting the contextual nature of the encodings they provided.

This contributes to the field of scientometrics and technology analysis, overcoming the gap in literature related to the analysis of patents for understanding the technological change. While interesting to practitioners, especially policy makers and companies, by supporting them in patent analysis to recognize emerging technologies, map technological convergence or investigate the content of a patent.

Moreover my analysis is relevant to the understanding of transformer based language models applied to diverse data sources and provides relevant insights from a diverse point of view. Specifically, it allows

to compare expertise driven approaches (lexicons and rules), with distributional methods employing the general knowledge obtained in pre-training.

This thesis focuses on analysing Transformer Based Language Models from several perspectives. I first test the amount linguistic knowledge encoded in BERT and particularly focus on investigating whether specific linguistic information is localized in identifiable sections of the embeddings. Doing so I find that there are specific entries relevant to the vast majority of linguistic tasks (Puccetti et al., 2021b). To investigate this phenomenon further I link it to outliers, weights that can disrupt the model when zeroed out. I analyse this relation from an extrinsic perspective, seeing how the removal of outliers affects the model embeddings and from an intrinsic one, looking at the parameters behaviour during training and linking it to the token distribution of language (Puccetti et al., 2022). Noticing that outliers arise during pre-training but affect the model also after fine-tuning I make a thorough analysis of different pre-training/fine-tuning schemes on models performance. Furthermore, to test these models also on diverse textual sources, I perform technological information retrieval from patents while comparing them with different methodologies to assess the improvements they provide (Puccetti et al., 2021a, 2023).

Recently, scaling Transformers to hundreds of billions of parameters and training on larger amounts of data (trillions of tokens), has led to breakthroughs in generative models. While such large models were not easily accessible during the development of these thesis, most of the findings either apply to models larger than those I experiment with or describe inspection methodologies that would be equally applicable to larger generative models. This holds for the applications to patents where, when newer methodologies will be developed, the same evaluation approach I set up in this work can be used to measure their performance.

The rest of this document is structured as follows, Chapter 2 goes through literature concerning transformer-based language models together with model inspection and technical information extraction from patents. The following one, Chapter 3 talks about the investigation of BERT linguistic knowledge encoding and localization, while Chapter 4 investigates outliers from several perspectives. Afterward, Chapter 5 covers applications, namely technical information extraction from patents and finally Chapter 6 ends the thesis by highlighting the relevant results and the conclusions drawn throughout the work.

CHAPTER 2

State of the art

Transformer Based Language Models have impacted all branches of Natural Language Processing (NLP) and have been doing so increasingly for a long span of time. Such an impact and relevance to a field (Rogers et al., 2020), makes it worth studying them, not only by measuring their functionality and investigating how well they address given tasks but also from a more fundamental perspective trying to understand their inner workings.

Therefore, this Chapter is composed of two parts. The first part is centered around the models themselves, I define what language modeling is and what transformers are, how the former can be tackled with the latter, and how this approach can transfer knowledge to most, if not all, problems in Natural Language Processing. The second part focuses on the application of these models to the extraction of technical information from patents. I describe why information extraction from patents is relevant, how it has been done before transformers, and how these models have been successfully used in this case as well.

2.1 Word Embeddings and Dense Representations of Text

Learning information from textual data has been a challenging task for a long time, before the recent rise of deep learning (Krizhevsky et al., 2012), Support Vector Machines (Mullen and Collier, 2004) have been among the most effective models.

More recently there has been a focus on creating representations of language (tokens/sequences), in dense fixed-length vectors, that carry textual information and can be used as features to tackle a large number of downstream tasks (Mikolov et al., 2013a; Pennington et al., 2014a).

This approach has grown in parallel with the development of artificial neural networks, for example, Recurrent Neural Networks (Goldberg and Hirst, 2017) such as Long-short-term-memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Cho et al., 2014) have been used effectively to encode textual meaning and tackle Natural Language Processing tasks.

One of the approaches to creating information-rich representations of text is training neural networks on language modeling (Bengio et al., 2003). This approach has been shown to be effective when performed with recurrent neural networks and scaled to large amounts of training data by Peters et al. (2018).

However, recurrent neural networks are difficult to train in some cases and have an information bottleneck given by their sequential nature. To address both these difficulties, Vaswani et al. (2017)

introduced a different architecture, called a Transformer, meant to learn sequential data, particularly textual, and they test it on machine translation tasks.

This approach works effectively on machine translation and it is later scaled and improved by Devlin et al. (2019) which uses Wikipedia (Foundation, 6 01) as a training set and later extended by (Liu et al., 2019b) scaling the same architecture without changes to training on a much larger dataset without exhausting the learning potential of the transformer used in both these studies. Devlin et al. (2019) train on language modeling with the specific goal of creating representations of text that can later be used to address a wide range of tasks, evaluate on GLUE (Wang et al., 2018) and (at the time of writing) reach state of the art by a large margin on all tasks.

Subsequently, this family of models has been applied to most natural language processing tasks and is currently the state of the art in the vast majority of natural language processing literature.

In the rest of this Chapter, I will thoroughly describe language modeling, the transformer architecture, and many of its variants. I then describe the attempts at interpreting the working of these models from linguistic and mechanistic perspectives. Finally, I will discuss their application in Named Entity Recognition in the patent domain.

2.2 Notation

To describe language modeling and the transformer architecture in detail let me establish a few notations that will be used throughout this Chapter. We refer to multidimensional vectors as *tensors* and the *shape* of a tensors indicates its size along each dimension. If v is a vector with k elements, v_i indicates its ith element and the $softmax$ of v refers to the vector such that

$$softmax(v)_i = \frac{e^{v_i}}{\sum_{j=1}^{k} e^{v_j}}$$

while the cross-entropy loss is computed as

$$CELoss(v, i) = -\log \frac{e^{v_i}}{\sum_{j=1}^{k} e^{v_j}}$$

. By t_i I indicate the tokens in a sequence, that is words or subwords, depending on the tokenization algorithm. Note that the available tokens, the *vocabulary*, is fixed once and after that new tokens are difficult to add.

2.3 Language Modeling

In its most general sense language modeling (Bengio et al., 2003) is the task of predicting a token (or a probability distribution over the vocabulary), given the preceding ones

$$\mathbf{P}(t_n|t_1, \ldots, t_{n-1})$$

however, the available context, which in this case is the preceding tokens, has been modified to address different goals.

2.3.1 Causal Language Modeling

To perform language modeling, the first step is assigning to each token, a trainable embedding, that is a vector of a given length d_k, this is done through an Embedding Layer (e.g. a single tensor) that maps each token in the vocabulary to its own embedding. In practice the Embedding Layer is a tensor with shape $(vsize, d_k)$, where $vsize$ is the vocabulary size, so that each row is the embedding of a unique token, let us call it $embedding$.

The embedding of each token is processed through the whole model[1], after the last layer a *Language Modeling Head* is used to compute the loss. A language modeling head is a linear layer (e.g. a tensor)

[1] I focus on the case of transformers, however, this is independent of the architecture used to compute the tokens' embeddings.

without a bias that maps each token embedding of length d_k to a vector of scores over the whole vocabulary allowing the computation of the language modeling loss. In practice, the *Language Modeling Head* is a tensor of shape $(d_k, vsize)$, and it maps the output of the model back to the shape of the vocabulary, let us call it $LMHead$. Note that the Embedding Layer and the Language Modeling Head in some sense have opposite roles and indeed, in some cases they are one the transposed of the other.

To describe how language modeling works more in detail, let S be a sequence $[t_{start}, t_2, \ldots, t_{n-1}, t_{end}]$ of tokens (each one represented by its index in the vocabulary) where the first t_{start} is a special token indicating the start of a sequence and t_{end} is a special token indicating the end of a sequence. To process them with the model they are first turned into embeddings (adding positional information) and in turn, these are processed by the model,

$$\begin{aligned} emb &= embedding([t_{start}, t_2, \ldots, t_{n-1}, t_{end}]) \\ out^m &= model(emb) \\ out^l &= LMHead(out^m) \end{aligned}$$

until the last layer. Looking at out^m, keeping the above notation, it is a tensor with shape (n, d_k) where n is the number of tokens in the sequence, and after the language modeling head, $out^l = LMHead(out^m)$ will have shape $(n, vsize)$ where $vsize$ is the number of tokens in the vocabulary. On the scores out^l the $CrossEntropyLoss$ is computed using as labels the shifted tokens, that is, the loss to optimize is

$$\begin{aligned} \mathcal{L}(t_n, y_n) &= CELoss(out^l_n, y_n) \\ &= CELoss(out^l_n, t_{n+1}) \end{aligned}$$

. A relevant detail when computing this objective is preventing unwanted tokens e.g. $t_{n+1}, \ldots$ to be used as features when predicting t_{n+1} itself, I will describe how this is achieved with transformers.

2.3.2 Masked Language Modeling

A different kind of language modeling, *Masked Language Modeling* is a similar objective that, given a token sequence, $S = [t_1, \ldots, t_{i-1}, t_i, t_{i+1}, \ldots, t_n]$ can be modeled as

$$\mathbf{P}(t_i | t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n)$$

where context from previous and subsequent tokens is used. In practice to learn this objective, t_i is replaced by a t_{mask} token, then tokens are processed as

$$\begin{aligned} emb &= embedding([t_1, \ldots, t_{i-1}, t_{mask}, t_{i+1}, \ldots, t_n]) \\ out^m &= model(emb) \\ out^l &= LMHead(out^m) \end{aligned}$$

and in a similar way to what is done for causal language modeling, the loss to optimize is

$$\begin{aligned} \mathcal{L}(t_i, y_i) &= CELoss(out^l_i, y_i) \\ &= CELoss(out^l_i, t_i) \end{aligned}$$

2.3.3 Contrastive Learning

While language modeling is effective in creating rich representations of text and images, a different challenge is creating representations of paired texts and images that are aligned with each other. This means creating representations such that image and text pairs with similar content are mapped to similar vectors. To this end, *Contrastive Learning* is an effective approach.

This objective (Radford et al., 2021) is computed over pairs of image and text and it consists of computing a cross-entropy loss using as input the similarity each image has with all the sequences in the batch, including the correct one, and as label the index of the image in the batch (and the opposite for each text representation).

To make this clearer, fix $S = [(im, t)_1, \ldots, (im, t)_n]$ a list of n image and text pairs such that that the ith image and the ith text represent the same thing. Given $model_{image}$ and $model_{text}$ two neural networks, I call $out^i = model_{image}([im_1, \ldots, im_n])$ the embedding of the images and similarly $out^t = model_{text}([t_1, \ldots, t_n])$ those of the texts, both are tensors of shape (n, d_k) where d_k is a model parameter. One can compute the matrix of similarity $SIM = out^i(out^t)^T$ that is $SIM_{i,j}$ is the scalar product between the representation of the ith image and jth text, then

$$\begin{aligned} \mathcal{L}_1(im_i, t_i) &= CELoss(SIM_i, i) \\ \mathcal{L}_2(im_i, t_i) &= CELoss(SIM_i^T, i) \\ \mathcal{L}_{contrastive}(im_i, t_i) &= \frac{\mathcal{L}_1(im_i, t_i) + \mathcal{L}_2(im_i, t_i)}{2} \end{aligned}$$

.

Contrastive learning is related to language modeling since, when computing representations of text, often Radford et al. (2021); Yu et al. (2022) the representation of each token in a sequence is obtained by preventing it to attend to previous tokens, in the same way as is done in causal language modeling.

2.4 Transformers

Neural networks can be used to learn any of the objectives described so far, among many architectures described over time, Vaswani et al. (2017) define the Transformer architecture. The key innovation in this work is *Multi Head Attention* and the description of the whole architecture that has later been used to tackle the vast majority of language learning tasks. First, I go through all the components of this architecture and their definition.

Attention is expressed as:

$$Attention(q_x, k_x, v_x) = softmax\left(\frac{q_x k_x^T}{\sqrt{d_k}}\right) v_x$$

where d_k is the size of the embeddings present in q_x, and k_x. This approach is based on the intuition that the scalar product between *queries* (q_x) and *keys* (k_x) acts as a smoothed look-up table that allows the model to "focus" on different parts of the sequences, *values* (v_x), it processes. I call $Att = softmax\left(\frac{q_x k_x^T}{\sqrt{d_k}}\right)$ the *Attention Matrix*. To make sense of how this computation works I start by looking at the dimensionality of the tensors involved in the case of textual input.

All the tensors q_x, k_x, v_x have shape (n, d_k) where n is the length of the embedded sequences and d_k is the dimension of the embedding used to represent each token. The entries of the attention matrix Att are computed as the scalar product between the embeddings of all pairs of tokens in the sequences q_x and k_x normalized by the $\sqrt{d_k}$ term. After the softmax computation, needed to normalize the rows to sum to one, the attention matrix is used to output a new representation for each token in v_x which is a weighted sum of the embeddings used to encode the input tokens.

Instead of regular attention, in transformers *MultiHeadAttention* is used, to allow for better contextual learning, based on the idea of applying attention separately on equally sized chunks of the inputs. To describe it, fix d_k the embedding size, nh the *number of heads*, and cs the *chunk size* such that $nh * cs = d_k$. The inputs q_x, k_x, v_x have size (n, d) where n is the number of tokens in each sentence and W^Q, W^K and W^V are matrices of learnable parameters such that $\hat{q}_x = W^Q q_x$, $\hat{k}_x = W^K k_x$ and $\hat{v}_x = W^V v_x$ all have shape (n, d_k), note that $\hat{v}_x$ can have a different shape (n, d_v) without changing anything else, I assume $d_k = d_v$ for simpler notation.

Calling $\hat{q}_x^i$ a tensor of shape (n, cs) such that

$$\hat{q}_x = concatenate(\hat{q}_x^i) \quad for \quad 1 \leq i \leq nc$$

where *concatenate* means joining tensors along the correct dimension. Similarly $\hat{k}_x^i$ and $\hat{v}_x^i$ are obtained and *MultiheadAttention* is defined as,

$$MultiheadAttention(q_x, k_x, v_x) = concatenate(Attention(\hat{q}_x^i, \hat{k}_x^i, \hat{v}_x^i)) \quad for \quad 1 \leq i \leq nc$$

.

Multi-head attention is meant to facilitate the learning of contextual information, indeed, the embedding of each token is the result of a (learned) weighted sum of the input embeddings of the previous layers' output. Stacking several layers on top of each other allows the encoding of fine-grained contextual information. This can be used to improve the quality of the representation of a single sequence, in this case, one talks about *self-attention* where all three q_x, k_x and v_x are equal. A different case is when q_x comes from one source while k_x and v_x from a different one, this is used to add relevant information from one sequence to perform actions on another, an example can be machine translation as is shown later.

Along with attention layers, transformers have other key components that are common to most architectural variations. First, attention layers are paired with a *Dense* layer, meant to further refine the representation of the hidden states computed for each token. Both attentions and feed-forward layers are complemented by residual connections (He et al., 2016) and layer normalization (Ba et al., 2016).

To use such an architecture to process text, each token in a sequence is embedded as a vector of trainable parameters of a fixed length, a positional encoding, of the same length, is summed (entry-wise) and finally all the stacked tokens composing a sequence are processed through a number of attention layers.

Developed by Vaswani et al. (2017) several deep neural networks tackling language processing tasks have shifted to this architecture, one of the first works has been BERT (Devlin et al., 2019), followed by several others (Radford et al., 2019; Liu et al., 2019b; Raffel et al., 2020) all of which train transformers on variants of language modeling.

As mentioned above one of the technical details that needs taking care of, when performing language modeling, is preventing the model from accessing part of the processed sequence. For example, the model can't use the representation of the token that it is currently predicting. This is always the case when performing language modeling, thus I describe how this is done in transformers.

The attention module is the only one that, to compute the representation of a given token, uses information from the remaining ones in the sequence. More specifically, the attention matrix, by weighting them, determines the extent to which other tokens can be used to compute each representation. Therefore, to prevent a given token to be used, one needs to zero out the entry in the attention matrix corresponding to that token.

The shape of the *mask* used to prevent information flow from unwanted tokens, changes shape depending on the type of language modeling. In causal language modeling, this will be a triangular matrix that allows each token to only attend to the preceding ones. In masked language modeling, the masked token should not be attended to.

The architecture described so far can be adapted to several settings, initially, it has been used in an *EncoderDecoder* fashion (Vaswani et al., 2017; Raffel et al., 2020) to perform machine translation. *EncoderDecoder* means that two similar transformers are used in parallel, the first, encodes a sequence in one language and is therefore called *encoder*, then the encoded sequence is given as input, together with its translation, to the second model, the *decoder*, performing the actual language modeling task.

More formally, let $S^1 = [t_1^1, \ldots, t_n^1]$ and $S^2 = [t_1^2, t_2^2 \ldots, t_{n-1}^2, t_n^2]$ be two sequences of tokens, and $model_1$ and $model_2$ the two transformers, then the following is done:

$$\begin{aligned} out^1 &= model_1(embedding_1(S1)) \\ out^2 &= model_2(out^1, embedding_2(S_2)) \\ out^l &= LMHead(out^2) \end{aligned}$$

And therefore the objective is

$$\begin{aligned} \mathcal{L}(t_n^2, y_n) &= CELoss(out_n^l, y_n) \\ &= CELoss(LMHead(out^1, embedding_2(S_2))_n, y_n) \\ &= CELoss(LMHead(model_2(out^1, embedding_2(S_2)))_n, t_{n+1}) \end{aligned}$$

where the output from the first model, out^1 is used to add information when computing the loss. The way the output from the encoder is passed on to the decoder is through cross attention, that is for some of the attention layers in the decoder, the query q_x is the encoding of S_2 from the previous layer (or the input itself), while the key k_x and value v_x are the encodings of S_1 as processed by the encoder. This allows the model to use the tokens up to x_n as features together with the whole S_1 sequence to predict x_{n+1}.

While this approach is effective for tasks involving pairs of texts, such as machine translations, encoder, or decoder only language models are also used with different goals. *Encoder* language models (Devlin et al., 2019; Liu et al., 2019b) are used to create rich representations of language. This is done by performing masked language modeling on large amounts of text. On the contrary *Decoder* language models (Radford et al., 2019; Brown et al., 2020) are used to generate text, by training with causal language modeling objective.

In parallel with the introduction of the transformer architecture, a novel training approach has been employed, pre-training/fine-tuning (Peters et al., 2018).

This approach consists of first training a model for a long amount of time on a language modeling task using a large unstructured dataset. After this is done, the model representations of text, that have only been trained to perform language modeling (of any kind), encode sufficient information to be useful in downstream tasks.

However, *fine-tuning* is still needed, this means that for any downstream task, with possibly more structured input text and labels, the model is trained one more time, this time with a loss specifically designed to tackle the task at hand. This second training is orders of magnitudes smaller and can be performed more rapidly with less resources.

2.4.1 Training Data

The rise of the pre-training/fine-tuning paradigm has brought along the need for ever larger datasets, starting from the whole Wikipedia (Foundation, 6 01), a well-structured dataset, up to corpora scraped from the web (Radford et al., 2019) in increasingly larger sizes (Raffel et al., 2020; Gao et al., 2021), including image and text datasets (Jia et al., 2021; Schuhmann et al., 2022).

Several neural language understanding tasks are used to benchmark new models, one of the most used is GLUE (Wang et al., 2018), this suite is meant to test language models on several aspects so that a model tailored for a given goal is not going to perform well on average on all tasks, while general purpose ones should.

The tasks composing GLUE are:

- CoLA (Warstadt et al., 2019) corpus of linguistic acceptability;
- SST (Socher et al., 2013a) Stanford sentiment treebank;
- MRPC (Dolan and Brockett, 2005a) Microsoft research paraphrase corpus;
- STSB (Cer et al., 2017) semantic textual similarity benchmark;
- MNLI (Williams et al., 2018) multi neural language inference;
- QNLI (Rajpurkar et al., 2016a) question neural language inference;
- RTE (Bentivogli et al., 2009) recognizing textual entailment.

To have an approximate idea of the difference between the pre-training data scale compared to the fine-tuning one, Table 2.1 shows the number of samples in all the training and validation sets in GLUE and in Wikipedia, which contains an order of magnitude more samples than the largest training set in GLUE. Since on GLUE recent models perform almost perfectly, a new suite of tasks has been built, to address GLUE limitations, SuperGLUE (Wang et al., 2019) composed of:

- CB (de Marneffe et al., 2019) a lexical entailment dataset;
- COPA (Gordon et al., 2012) choice of plausible alternatives;

DataSet	# train samples	# val samples
wikipedia	6078422	-
cola	8551	1043
sst2	67349	872
mrpc	3668	408
qqp	363846	40430
stsb	5749	1500
mnli matched	392702	9815
mnli mismatched	392702	9832
qnli	104743	5463
rte	2490	277
wnli	635	71

Table 2.1: *GLUE and Wikipedia (01/05/2020) dataset number of samples (train and validation).*

- MultiRC (Khashabi et al., 2018) multi sentence reading comprehension;
- RTE (Bentivogli et al., 2009) recognizing textual entailment, kept from GLUE;
- WIC (Pilehvar and Camacho-Collados, 2019) word in context;
- WSC (Levesque et al., 2012) Winograd schema challenge, kept from GLUE;
- BoolQ (Clark et al., 2019a) yes/no question answering;
- ReCoRD (Zhang et al., 2018) reading comprehension with commonsense reasoning;
- Winogender: (Rudinger et al., 2018) winogender schema diagnostics.

A dataset not included in these suites but often used alongside them is SQuAD (Rajpurkar et al., 2016b) an extractive question-answering dataset, composed of context/question/answer triples meant to test reading comprehension, which has been later extended (Rajpurkar et al., 2018) with context/question pairs where some contexts don't contain an answer to make the benchmark more realistic.

Moreover, with generative models rapidly improving in performance, the benchmark suites have grown in complexity and size, Srivastava et al. (2023) describe a collectively gathered suite with over 200 challenging benchmarks meant to be updated over time.

2.4.2 Architecture Variants

After the initial transformer implementation, several variants have been proposed, with the goal of addressing some of the limitations of the original architecture.

As described above the maximal length of the sequences that can be processed by a language model is fixed at the model definition and the computational cost of the attention layers is quadratic in the context length. To address this issue, Dai et al. (2019) define a training strategy that allows learning longer context in a recurrent way. The main idea is that the representations of a sequence used as context encodes both the sequence being processed and the previous one. This allows the model to retain information from the previously processed sequences up to a given point, in practice this model does show better performance on tasks that need a longer context.

A different approach to allow a model to process longer sequences without increasing computation costs is to enforce attention patterns to make the attention layers have a less than quadratic cost, Beltagy et al. (2020) do this by having local and global components in the attention patterns so that each token attends to both further and closer inputs in the rest of the sequence while only keeping $\mathbf{O}(n)$ non-zero entries in the attentions matrix. A similar approach is also used by Zaheer et al. (2020) where also theoretical limitations for sparse attention computational costs are derived.

A complementary approach to improving language models has been pursued by changing the objective while keeping the language modeling core. Variants of masked language modeling have been proposed. In particular, Zhang et al. (2020a); Lewis et al. (2020) introduce the concept of denoising language, consisting of replacing a given token t_i with a different one chosen according to different schemes instead of the mask token t_{mask} and adding noise to the embedding of the token used as a replacement, making language modeling harder. This allows to tailor language modeling to the task at hand and makes training shorter, in particular, Zhang et al. (2020a) use a denoising approach with the goal of performing summarization. Lewis et al. (2020) make the learning process more effective by training an encoder-decoder model mixing language denoising together with causal language modeling.

Clark et al. (2020) improve language modeling by first training a small language model, a generator, and later using the token probability it predicts to build a challenging language modeling scheme. This means that each token t_i that in masked language modeling would be replaced with t_{mask}, is instead replaced with t_i^{gen}, a token chosen according to the distribution output by the generator for the position i in the sequence. This makes the training of the model faster as replacement tokens are chosen to be more challenging than random.

To increase the representational power of language models, a different component that is improved are positional encodings, these are part of the input that is used by these architectures to add positional information to the embedding of a given token so that sequences such as "the happy dog" and "the dog happy" can be interpreted as different even though the tokens composing them are the same. Originally, Vaswani et al. (2017) describe two possible positional embeddings, a learnable one, that consists of a vector of trainable parameters for each position in a sequence that is added to the embedding of each token and a static one that is obtained by adding a vector with sinusoidal entries so that the model is evaluated at different points when computing the forward pass and hence can leverage this information to distinguish among same tokens in different positions.

However, both these positional embeddings only have global information, that is they allow the model to learn the position of a token in a sentence but not relative to other tokens. To add the possibility of recursively adding information from previous sequences, Dai et al. (2019) add the concept of relative positional encodings. This is meant to only let the model encode information about the distance $i - j$ assuming $i > j$ between a query $(q_x)_i$ and a key $(k_x)_j$. This is done by refactoring the computation of the attention matrix to embed this information instead of the absolute position, thus allowing the model to recursively encode information about past sequences. After being introduced in the context of language modeling by Dai et al. (2019) relative positional encodings have been reused and further studied by others (Yang et al., 2019a).

An approach that uses several of these insights is implemented by He et al. (2021) who use separate attention for embeddings and positional encodings and add a way to decode absolute positional information directly into the language modeling head.

A parallel attempt at improving transformer-based language models has focused on reducing the computational cost of these models. This has been done in several ways. Perhaps the most straightforward way is adopted by Lan et al. (2020a) which reduces a model composed of several layers (e.g. 12 for BERT) to a smaller one composed of a single layer which accumulates an equivalent number of steps before performing the backward pass. This allows to dramatically reduce the model size, although at the cost of performance.

A different approach toward the reduction of computational costs can be achieved via pruning, which means zeroing out a share of the parameters to reduce the cost of running the model. Sanh et al. (2020) achieve this via movement pruning, that is, removing those weights that change the most during training. Prasanna et al. (2020) show that zeroing out a large share (up to 40%) of the model parameters, after it has been pre-trained, does not harm the model performance on downstream tasks.

While helpful to make a large model computationally cheaper, these techniques do not allow to train a large model less expensively, they only allow to use model at inference time for a smaller cost. Similarly, model quantization techniques allow using extremely large models with smaller computational needs (Dettmers et al., 2022; Dettmers and Zettlemoyer, 2023; Wei et al., 2022b) these works allow to run models with up to 175 billion parameters on relatively small gpus.

An alternative solution is based on the idea of creating smaller models retaining larger ones' performance. Distillation works towards this goal and consists in using a larger model as a "teacher" and a

Model name	Reference	model size
BERT-base	(Devlin et al., 2019)	$\approx 110M$
BERT-large	(Devlin et al., 2019)	$\approx 440M$
GPT2	(Radford et al., 2019)	$\approx 1.5B$
T5	(Raffel et al., 2020)	$\approx 11B$
GPT3	(Brown et al., 2020)	$\approx 175B$
PaLM	(Chowdhery et al., 2022)	$\approx 640B$

Table 2.2: *Model size in number of parameters (M: millions, B: billions) for a few notable models.*

smaller one as a "student". The smaller student model is trained with a loss that is based on language modeling but adds a component aimed at mimicking the teacher model distribution over vocabulary (Sanh et al., 2019).

A different result that allows making attention computation more efficient comes from Dao et al. (2022), who, through a technical analysis of the costs of each component of attention computation ranging from data loading to gpu efficiency, devise a cheaper and exact algorithm to compute attention. Similarly, Choromanski et al. (2021) split the attention computation into blocks and using several combined hashing tricks, describe a faster attention computation method that retains most performance.

A different, structural, limitation of how language modeling is performed (not only with transformers) is the need to predict the next token over the whole vocabulary. The final softmax normalization makes the resulting distribution concentrated on a few tokens (Chang and McCallum, 2022), making the learning of the token distribution harder. This is addressed by computing the softmax of partitions of the vocabulary and then aggregating the results (Yang et al., 2018) and this approach can be applied to transformers too (Yang et al., 2019b). This change on the original transformer architecture is one of those that when scaled to a larger model size does not degrade performance compared to the regular architecture (Tay et al., 2022a).

Despite the many variants developed, some of them have been shown to only match the original architecture at a smaller scale while for larger size such changes harm performance on downstream tasks (Tay et al., 2022b).

Using transformers for language modeling has made it possible to train very large models with increasingly positive results, (Raffel et al., 2020; Brown et al., 2020; Hoffmann et al., 2022; Chowdhery et al., 2022).

Beyond language modeling and text representation, the same architecture, surprisingly also with similar training techniques, has proven useful in visual tasks (Parmar et al., 2018; Dosovitskiy et al., 2021).

Recently, other fields besides image and text processing have employed this architecture to tackle domain-specific tasks such as life-science (Rao et al., 2021) and speech processing (Baevski et al., 2020).

While several new models have been recently developed, to understand the pace at which the size of such models is growing it is sufficient to look at a few, Table 2.2 shows the size of some of the most notable models and reports that over three years there has been a growth factor of 10^3 in model sizes.

2.5 Probing

One of the common features of all language models reported so far is the size of the architectures, indeed, the number of parameters is larger than 10^5 in the vast majority of those mentioned.

This makes interpreting their output and assessing the mechanisms behind their good performances challenging. Starting from this issue, there has been a coral effort towards interpreting these models from a vast range of perspectives, from a purely linguistic perspective (Conneau et al., 2018; Vig and Belinkov, 2019; Hewitt and Manning, 2019a), from an explainability perspective (Ribeiro et al., 2016; Lundberg and Lee, 2017; Ribeiro et al., 2018), from a mechanistic perspective (Kobayashi et al., 2020; Hohman et al., 2020) among others (Rogers et al., 2020).

Investigating their inner mechanism and the linguistic knowledge they implicitly encode (Belinkov

and Glass, 2019) has affirmed as a solution to interpret language models. One of the most common approaches is based on the development of *probes*, i.e. supervised models trained to predict simple linguistic properties using the word/sentence embeddings of a pre-trained model as training features.

Adi et al. (2017) use this methodology to compare different sentence embedding methods. They investigate which, among several that encode tokens in fixed-length vector representation, is most effective in different tasks including token length and word order prediction, with a focus on making a cross-analysis between the vector dimensionality and the task for which each representation is best suited.

While using similar approaches, Belinkov et al. (2017) focus on evaluating the relevance of neural network depth, the number of layers, on the model's ability to encode linguistic information. To do this they use part-of-speech tagging and semantic tagging tasks to probe the model's hidden states at different depths and find that this information is present to different degrees at different depths.

Using similar auxiliary (probing) tasks, Zhang and Bowman (2018) compare language modeling to neural machine translation pre-training. investigating properties such as syntactic and part-of-speech tagging, they find that models pre-trained on language modeling are better at producing embedding to infer such properties than those trained on neural machine translation.

The probing approach has later been used to test models producing sentence representations in increasingly complex tasks. Conneau et al. (2018) probe several models in a suite of tasks ranging from simpler properties, e.g. sentence length prediction, to increasingly more complex syntactic ones, e.g. depth of the dependency tree, and semantic ones, e.g. verb tense prediction. They find that models encoding context work better than those that do not. Moreover, they find that different architectures trained with the same objective result in representations with different properties underlining the importance of the architecture alongside the learning objective.

Miaschi et al. (2020) probe BERT for a wide range of sentence-level linguistic tasks at different model depths, find a positive correlation between the amount of information that this model can encode at a given layer and how effectively the hidden state output by that layer can be used to perform downstream tasks. Moreover, they find that while this is true for pre-trained models, after performing fine-tuning, this ability is largely lost.

These studies demonstrate that neural language models are able to encode a wide range of linguistic information in a hierarchical manner, in particular, Blevins et al. (2018) show that pre-trained language models encode varying amounts of hierarchical knowledge at different depths. It is relevant that they find this behavior in models pre-trained on dependency and semantic role labeling, tasks supposed to inherently encode such knowledge as much as in models pre-trained on machine translation and language modeling, which, at least intuitively, should incorporate such knowledge to a lower degree.

Similarly, Jawahar et al. (2019) show that several properties of language can be extrapolated from BERT hidden states at different depths, moreover they show that long-distance dependency is stored at deeper layers and that the sentence-level representations provide more accurate sentence-level information.

Tenney et al. (2019b) focus on defining tasks meant to quantify the amount of sentence-level information and use them to compare contextual and non-contextual representations of text. They conclude that, not only models that can learn in-context information perform better but also that some of their results can not be accounted for by non-contextual approaches, proving from an experimental perspective that it is the contextual component of the architectures that contributes to improvements.

Focusing on architecture, rather than on contextual versus non-contextual representation, Liu et al. (2019a) quantify differences in the transferability of individual layers between different models, showing that higher layers of RNNs (ELMo) are more task-specific (less general), while transformer layers (BERT) do not exhibit this difference in task-specificity.

Similarly, making use of information regarding model architecture while taking on more challenging tasks, Hewitt and Manning (2019b) show that from contextual representation it is possible to extract the dependency parse tree. To do this they need to devise a more complex probe, that maps geometric information of the embedding space to the dependency trees structure, by exploiting this they are able to show that tokens are mapped in a way that encodes the information about the dependency tree without it being enforced through the training objective. They also show that this same approach does not work to reconstruct the dependency parse tree from static token representations.

Hewitt and Liang (2019) propose to improve probing tasks by measuring how well they perform on identifying linguistic properties while maintaining low performance on control tasks, in order to avoid

assessing a mixed signal. This leads them to conclude that in some cases, simpler classifiers, which can not capture the full information present in the representation of language, output by language models, are better for probing.

While probing has mostly been used as an empirical approach to measure the amount of linguistic information present in language models, Pimentel et al. (2020) formalize this concept from an information theoretical perspective and argue that while other works favor simpler probes, the best performing one is always the correct one.

To devise an approach to study language models from a data-centered perspective, McCoy et al. (2019a) compose a dataset of examples with misleading heuristics and show that in general models perform almost randomly on these samples, unless they are shown some in their training data, in which case they learn such anti-heuristics with more than 90% accuracy.

2.6 Geometry of the Embedding Space

Investigating the amount and quality of linguistic information present within the representations of text that language models provide, reveals interesting properties of these models, however, it gives limited information on their inner workings. A different perspective, still only focusing on the models' output, consists in investigating how they shape the representation space from a geometric point of view.

There is a focus on trying to address the anisotropic nature of embeddings produced by language models, which means that tokens' representations are aligned along a few directions in the high-dimensional space where they are embedded.

The reason to work towards isotropic embeddings is that highly nonisotropic ones have lower representational power. Indeed, making the average similarity among representations higher, makes it harder to differentiate between similar and dissimilar tokens/sentences.

To address this Liang et al. (2021) add learnable parameters aimed at making the prominent directions in the representations of a language model smaller. Improving performance in tasks that need fine-grained word representations such as word similarity.

A different point of view on this phenomenon is investigating how it affects fine-tuning of language models. It is shown that fine-tuning on specific tasks increases the anisotropicity of the representation space (Rajaee and Pilehvar, 2021) challenging the claim that anisotropic representations are desirable.

Gao et al. (2019) prove formally and experimentally that the anisotropicity is due to the language modeling loss, however, they only prove it in the case of *weight tying*, that means when the token embedding matrix and language modeling head share parameters, which is not the most general setting. To understand the geometric properties of the representation space, for classification tasks, it is interesting to inspect examples close to the decision boundary, since they are the most affected by an anisotropic embedding space.

McCoy et al. (2020) show that fine-tuning several times the same model on differently shuffled data leads to different kinds of biases that can be inspected also among the examples close to the decision boundary. Not only among different runs of the same model but also among different models based on the same principles, there are strong differences in the geometry of the embedding space.

Ethayarajh (2019) show that the embeddings generated by three similar architectures Elmo (Peters et al., 2018), BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019b) (the last two only differ in training data) showcase different behaviors. Moreover, they point out differences among the embeddings generated at different model depths and show that context can motivate at least part of the success of contextual word representations. Timkey and van Schijndel (2021) show that the anisotropicity of the representation space is caused by few dimensions, find that there is statistical significance to this effect and measure the impact of such dimension on word/sentence similarity tasks.

Moreover, Puccetti et al. (2021b) show that these dimensions are tightly related to the model's ability to encode a large share of linguistic information into the embeddings they generate. Similarly, Torroba Hennigen et al. (2020) devise a probing strategy that investigates parameter relevance to given linguistic properties of the encoded text and find out that few parameters account for a large share of such information.

As I have shown there is a large stream of literature investigating representations created by large language models, their linguistic skills and the geometry of their representation space, focusing on their

output. However, such phenomena are rooted in the model parameters and a closer analysis of intrinsic properties reveals model behaviors that are harder to inspect "from the outside". Perhaps the most spread approach that has been used is studying attention (Rogers et al., 2020). Indeed, this network component, as described in Section 2.4, is naturally prone to interpretation since it is based on the idea of building the representation of each token at a layer based on a weighted sum of the representations output by the previous one.

Kovaleva et al. (2019) investigate the most common patterns in attention, come up with five high-level ones and try to relate these patterns to specific properties of language. Before using it as an approach to interpreting model output Kobayashi et al. (2020) measure the amount of information that can pass through attention layers and conclude that the attention mechanism can encode more information than a simple look-up table, this is interesting since attention can be seen as a "smoothed" look-up table.

Baan et al. (2019) train two identical language models adding to one a constraint on attention, this allows to inject desired behaviors into the model and shows that the specific constraint on attention indeed does help in compositionality tasks, for which the constraint is designed, bridging the interpretation of model behavior and the attempt to inject desired properties.

Jain and Wallace (2019) challenge the idea that attention can be used as an explanation method arguing that the term *explanation* has a stronger meaning than the simple token relevance within a sentence and that therefore only looking at attention by itself is not a sufficient explanation technique.

2.7 Individual Neurons and Outliers

In an effort to investigate language models' inner workings, other approaches look for information encoded in single neurons, trying to identify if the model can encode specific information in localized weights.

Qian et al. (2016) find that specific parameters are relevant to specific linguistic tasks and that the differences are based on the complexity of the task, ranging from lower-level knowledge such as part-of-speech-tagging to higher-level ones such as dependency parsing.

Bau et al. (2019) show that there are properties that all models trained for neural machine translation share. They use this finding to identify parameters relevant to machine translation by correlating their values in different models and finally show how modifying such neurons allows to actionably control the translation results.

Dalvi et al. (2019) propose two methods, *Linguistic Correlations Analysis* and *Cross-model correlation analysis*, to study whether specific dimensions learned by end-to-end neural models are responsible for specific properties. For instance, they show that open class categories such as verbs and location are much more distributed across the network compared to closed class categories (e.g. coordinating conjunction) and also that the model recognizes a hierarchy of linguistic properties and distributes neurons based on it.

Lakretz et al. (2019), instead, propose a detailed study of the inner mechanism of number tracking in LSTMs at the single neuron level, showing that long-distance number information (from the subject to the verb) is largely managed by two specific units.

One of the properties of a language model parameters that is studied are *outliers* (Kovaleva et al., 2021). Outliers are specific parameters of a language model, present in the final component of each layer, generally either a Normalization layer (e.g. for BERT which uses post normalization) or a Linear layer (e.g. for GPT2 which uses pre normalization), that when zeroed out drastically impact the model performance on both language modeling and downstream tasks. Zeroing out only 48 parameters in a model with approximately 110 million can degrade accuracy on given tasks by up to 30% (Kovaleva et al., 2021).

This is not the case in general, for randomly picked parameters there is a rather opposite trend, once trained, up to 40% of BERT's weights can be zeroed out without harming performance (Prasanna et al., 2020). Similarly, Dalvi et al. (2020) show that for specific downstream tasks even larger shares of parameters are redundant, and they devise fine-tuning adaptations that allow to remove up to 90% of the parameters in certain tasks. This makes the outlier phenomenon relevant showing that these parameters play a special role in the model architecture.

Puccetti et al. (2021b) show that such parameters are relevant in a large number of linguistic probing tasks and using intermediate checkpoints (Sellam et al., 2022), Puccetti et al. (2022) show that such outlier parameters arise during pre-training and their removal increasingly affects the model behavior as training

proceeds. Outliers are also tightly related to the anisotropic representations generated by language models, since they are close to the output of each layer, they contribute strongly to the properties of the hidden states output by the model.

It is noteworthy that they behave like an emergent property (Dettmers et al., 2022). This means that these parameters are tightly related to the model size and they become more consistent at a given model size while absent earlier. A similar phenomenon is observed by Luo et al. (2021) which inspect BERT and RoBERTa to find out that outliers appear in the embedding layer, the one closer to the network input.

After describing language modeling and the architecture of transformer-based language models, their variants, limitations and the attempts at understanding their inner mechanisms, I turn to show how they have been applied to perform information retrieval from technical documentation, in the remainder of this Chapter I focus on the extraction of technical entities from patents. To understand how the introduction of these models has had an impact for this task, I first describe how previous natural language processing techniques have been applied to it in the past. Then I show that transformers have been adopted in this context as well, that they improve performance and argue why they can achieve that.

2.8 Applications to Patent Analysis

Patent analysis has been involved in both academic and industrial disciplines for different purposes. Scholars are generally interested in the analysis of patent data for identifying and studying new radical innovation and paradigm shifts. Whereas, industry actors are focusing also on incremental innovation, namely technical modifications on existing technologies (Sternitzke, 2010).

Patent analysis techniques are capable of performing a wide range of tasks, relevant from both legal and managerial perspectives (Liu et al., 2011; Yoon and Kim, 2011). Abbas et al. (2014) review the literature on patent analysis identifying several purposes of the efforts in analyzing intellectual property, among which there are novelty identification, technological forecasting and technological road map. Prior and current works have relied on two main approaches for the analysis of big amounts of patent data: bibliometric analysis and text mining, as affirmed in several works (Abbas et al., 2014; Li et al., 2019; Small et al., 2014).

The former comprehends a wide range of techniques focused on the analysis of structured data, such as patent assignee, inventor, citation, International Patent Classification (IPC) or Cooperative Patent Classification (CPC). The literature provides a plethora of indicators developed using patent meta-data for a specific purpose. For example, the number of patent applications, backward/forward citations and the number of non patent literature (NPL) citations are used for analysing technological diffusion (Chang and Fan, 2016; Magee et al., 2016). Co-inventors, co-citations, NPL citations and IPC co-classification are metrics, based on meta-data, for identifying technological convergence (No and Park, 2010; Karvonen and Kässi, 2013; Cho and Kim, 2014) or emerging technologies (Small et al., 2014; Breitzman and Thomas, 2015; Kyebambe et al., 2017; De Rassenfosse et al., 2013).

Despite the large adoption by academics and practitioners, bibliometric patent analysis has limitations. Citation analysis is able to capture prior art but lacks the detail to reflect the technical content of each patent (Kuhn et al., 2020). On the other hand, patent classification reflects the subject matter of the document but patent classes are too broad to capture the detailed technical content of each document (Righi and Simcoe, 2019). Arts et al. (2021) illustrate and validate the improvement of text mining metrics over traditional measures (based on bibliometric data) for capturing the innovation phenomenon using patents.

Natural Language Processing is applied to virtually all contexts where written documents are available. Among its many application, Text Mining aims to automatically extract the information contained in written text. A branch of text mining for a range of applications, such as machine translation, document classification, question answering, named entity recognition (Pennington et al., 2014b).

One of the simplest algorithms in NLP is the Bag-of-Words (BoW), that is a representation of a set of documents (namely *corpus*) that describes the occurrence of words within each document. Hofmann et al. (2019) use BoW for measuring the text-based similarity between patents in order to generate technology-related network data that retraces elapsed patterns of technological change. An improvement of BoW model is weighting the word occurrences with the commonly used term frequency inverse document

frequency (tf-idf) (Salton et al., 1983). In Niemann et al. (2017), a tf-idf BoW aims to structuring patent text to identify patterns of change over time.

This representation of the textual content of a patent is not useful for capturing the latent semantic of the document. Latent Semantic Analysis (LSA) and Singular Value Decomposition (SVD) are two of the dimensionality reduction techniques used in the literature for overcoming this issue. In Magerman et al. (2010), the authors involve LSA for grasping similarities between patent and publication text documents. In Park et al. (2015), a patent similarity index based on LSA is developed for exploring potential R&D collaboration partners. SVD is involved by Han and Sohn (2015) to provide a concept of distance between a document and its backward or forward cited patents in terms of claims.

While NLP methods mentioned so far aim to compare the similarity between a set of patents, Latent Dirichlet Allocation (LDA) is employed in literature to extract topics and technological trends. LDA is a popular model used in the field of information retrieval from text corpora, developed in Blei et al. (2003). The primary benefit of LDA is predicting the future technological topics of specific firms (Suominen et al., 2017). In Kim et al. (2015), the authors apply LDA algorithm to identify a vacant technology clusters using patent documents. In Song and Suh (2019), the authors aim at identifying the convergence trajectory for safety technology development via topic modeling techniques.

Recently, the attention of NLP literature has focused on word embedding algorithms, these are able to represent words as dense vectors of real numbers. Unlike LSA and LDA, which also focus on estimating continuous representations of words, word embedding methods use an artificial neural network to represent words. In Mikolov et al. (2013b), the authors show that word embedding outperforms the LSA algorithm for preserving linear regularities among words and have greater computational efficiency than LDA with large text corpora (Mikolov et al., 2013a). Xu et al. (2019) use word embedding to automatically extract technical intelligence from news. Word2vec (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014b) are two recent and popular word embedding methods for analysing patent documents. In Lee et al. (2020a) the authors use word2vec developing a product landscape analysis in order to identify potential technology opportunities across multiple domains. In Trappey et al. (2019), a system based on word2vec is used for retrieving the best related patents of a target document aiming at analysing patent evolution in solar power technology. In Sarica et al. (2020), GloVe is applied to vectorize the patent terms and establish their relationships in a unified vector space for representing the technology semantic network. In Li et al. (2018), the authors propose a machine learning method for patent classification based on word embedding.

2.9 Technology Extraction From Patents

Named Entity Recognition (NER) consists in detecting lexical units in a word sequence that refer to a predefined entity, thus determining what kind of entity it is (e.g. persons, locations, organizations) Figure 2.1.

This is an Information Extraction technique that aims at recognizing unstructured text information units (e.g. foods, person names, companies, geographical entities) (Nadeau and Sekine, 2007).

The most successful NER systems focus on meaningful text representation trough word embeddings (Gildea, 2001; Piskorski and Yangarber, 2013; Cer et al., 2018; Liu et al., 2020).

Information about the entity to which a word belongs, can provide crucial, although shallow, semantic information for tasks such as question answering (Abujabal et al., 2018; Blanco-Fernández et al., 2020), topic disambiguation (Fernández et al., 2012) or detection (Lo et al., 2017) and elements relationships identification (Sarica et al., 2020).

Technical documentation and patents are a rich source of valuable technical information (Abbas et al., 2014). Named Entity Recognition is an effective methodology for the extraction of such information (Park et al., 2013; Chiarello et al., 2019; Sarica et al., 2020; Liu et al., 2020) while from a different perspective the complex nature of technical writing provides an interesting test bench for this methodology.

Within the context of patents there have been mainly three NER methods used in the literature:

1. *lexicon-based NER* uses knowledge base resources (or lexicons in jargon), consisting of lists of known instances (Pawar et al., 2012), to map mentions of entities within texts to knowledge base resources (e.g. Wikipedia). It is a fast and accurate approach for NER. The lexicon-based NER is

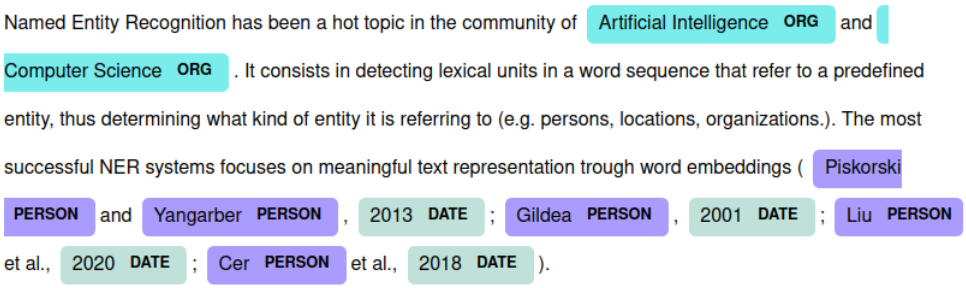


Figure 2.1: *Example of NER tagging.*

a semi-supervised method since list creation usually requires manual effort and domain-specific knowledge;

2. *rule-based NER* uses regular expressions and morphosyntactic information to express knowledge based systems able to extract a certain type of entity (Jiang et al., 2011). This system is unsupervised and top-down, and falls into the class of expert systems;

3. *distributional-based NER* uses manually annotated documents (training set) to train machine learning algorithms (Quinlan, 1986). These methods are the state-of-the-art in many fields of Natural Language Processing, but they require the effort of collecting and annotating large data-set.

The key metrics used in the evaluation process are precision, recall and f-score.

One of the most challenging properties to inject in all the NER systems described is accounting for context. Indeed, whether a word is or not part of entity is to a large extent determined by the context where it appears.

Transformers overcome this issue factorizing a word or a phrase based on the textual context where it appears. One of the earliest models, BERT (Devlin et al., 2019), has been successfully applied in this context too. In Caragea et al. (2020), a BERT-based model is used on a dataset of patent abstracts and patent applications for building a taxonomy of Financial Technology (FinTech). Beltagy et al. (2019) fine-tune a BERT model based on scientific articles, called SciBERT, for processing the text of this huge amount of data makes available from scientific communities. The model is largely used in literature for many purposes, also related to the study of technological phenomena Zhang et al. (2021).

In Lee and Hsiang (2020), the authors fine-tune with about 2 millions patents a pre-trained BERT model, called PatentBERT by the authors. Specifically, they use this model for classifying the patent documents based on the International Patent Classification. Recently, PatentBERT is adopted as a comparison in various patent analysis-based studies. For example, Hain et al. (2022) develop a new method to create vector representations of patent text for measuring the patent-to-patent similarity, and compare their performance with PatentBERT model. Similarly, Choi et al. (2022) propose an automated patent classification model for patent landscaping and compare the classification performance of their model with PatentBERT.

In the literature, there are various works that are able to achieve good results for the identification of named entities. Lee et al. (2020b) aim to recognize biomedical entities for obtaining a f1 score of 89.61 for diseases, 94.26 for drugs and 78.58 for genes and proteins. In a similar manner, Fan et al. (2020) achieve a level of f1 equals to 75.37 for the extraction of events and location names and both techniques used contextual transformer-based language models.

On the contrary, the literature on NER systems for technology extraction is poor. In Jang et al. (2021), the authors aim at structuring textual information as a cornerstone of a lexical database for technology-related information, based on the patent data. Similarly, Sarica et al. (2020) develop a technology semantic network (TechNet) using natural language processing techniques to extract terms from massive patent texts. TechNet supports a wide range of applications, e.g., technical text summaries, search query predictions and relational knowledge discovery in the context of engineering and technology. Furthermore, both works

construct a lexical database of technology-related words but they do not recognize a given technology in the patent text.

Hossari et al. (2019) focused on the extraction of Artificial Intelligence technologies. However, the authors give only qualitative criteria for evaluating their results. The state-of-the-art is mainly focused on the identification of emerging technologies and not established ones, exploiting patent classification (i.e. IPC code classes) (Kay et al., 2014; Gustafsson et al., 2015; Song et al., 2017) or using textual data with greedy methods, extracting generic terms and not only technologies.

For example, Ranaei et al. (2020) analyse the emergence of technologies using three text-based approaches: tf-idf metrics for capturing technological changes, LDA for evaluating the emerging topics, text-based score developed by Porter et al. (2019) and Carley et al. (2018). The authors found that the three different methods provide somewhat distinct perspectives improving the understanding of technological change.

While a training set for entities like chemical molecules and biomedical terms has been built to train corpus-based classifier, this is not true for technologies. This forces to manually evaluate the precision of the NER system, reviewing the list of extractions to filter out non-technologies. For what concerns recall, in absence of a training corpus, the grain of the extracted entities is the key element to consider for evaluating it. Linguistically, one can refer to technologies using an abstract wording (e.g. *device* or *system*) or specific wording (e.g. *water proof smart watch* or *visual software modeling editor*). For a balanced identification of technological content in a patent corpus, it is important to avoid both the extraction of too generic and too specific technologies. Taking into account too generic technologies causes the risk that no technological change and evolution pattern would be identified by the patent analysis. For what concerns the extraction of too fine grained technologies, it is important to avoid collecting sub-systems instead of technologies.

In a recent paper, Giordano et al. (2023) involve lexicon-based and rule-based system for identifying technologies from a set of 300,000 patents related to C4ISTAR, a defence related domain, for analysing the convergence phenom. They reached a precision of 35.39%, collecting about $1,000$ different technologies. However, the purpose of the authors is different from mine for three main reasons. First, the authors focused on the study of technological convergence phenomena and not on developing a NER methods able to extract technologies from the text. This different purpose of Giordano et al. (2023) leads to a lack of measurement testing their NER method. In fact, the authors only evaluated the precision of their method without calculating the recall or providing information about the time they spent to recognize technologies from the text. Second, Giordano et al. (2023) avoid to use state-of-the-art NER system, namely Transformer models method, based on contextual word embeddings, but they only applied lexicon-based and rule-based NER. As discussed before, these methods suffer some limitations, related to the recall measurement.

2.9.1 The Concept of Technology

As suggested by Nadeau and Sekine (2007), the use of NLP techniques for technology identification and mapping requires a formal definition of technology, precise enough to allow discerning these entities and comprehensive enough to include all of them. Therefore, I review several definitions from dictionaries and literature, reported in Table 2.3, to highlight the main characteristics of this entity and to identify what are the elements for which an entity can be classified as a technology.

The reported definitions in Table 2.3 convey different aspects, each relevant to this concept, the most insightful elements are:

(A) the technical means or in general the technical systems (Wikipedia, Waight 2014; Volti 2005; Ramanathan 1994);

(B) the anthropocentric view, related to the creation and the use of technology by human-kind (Wikipedia, Carroll 2017; Waight 2014; Volti 2005; Ramanathan 1994);

(C) the application of knowledge and science thanks to the skills of many individuals, who cooperate themselves to develop a technical system (Cambridge dictionary, Carroll 2017; Volti 2005; Ramanathan 1994);

(D) the purpose to solve practical problems and to perform a function (Carroll 2017; Waight 2014; Volti 2005).

To address the need for an operative description enabling to define a scope and perform an efficient and strict selection of entities identifying a technology I attempt to distill all the multiple perspectives aforementioned into a comprehensive summary, as follows:

Definition 1. *A* technology *is a technical mean or in general a technical system (A) created by human-kind (B) through the application of knowledge and science (C) in order to solve a practical problem or perform a function (D).*

With this definition I do not have the ambition to define in a complete and exhaustive way such a broad and complex concept. The aim is indeed to clarify the main characteristics of a technology in order to be able to distinguish the relevant information among the amount of data extracted with NLP techniques from patents.

Source	Definition
Cambridge dictionary	(The study and knowledge of) the practical, especially industrial, use of scientific discoveries.
Wikipedia	Technology ("science of craft", from Greek $\tau\epsilon\chi\nu\eta$, techne, "art, skill, cunning of hand"; and $-\lambda o\gamma\iota\alpha$, -logia)[1] is the sum of techniques, skills, methods, and processes used in the production of goods or services or in the accomplishment of objectives. [...] It can be the knowledge of techniques and processes, or it can be embedded in machines to allow for operation.
(Carroll, 2017)	Technology is "something that is always inherently intelligent enough either to function and to be used to function; anything devised, designed, or discovered that serves a particular purpose; [and] the knowledge that is used for a purpose, without itself necessarily being translated into something physical or material that does (e.g., instructional methodologies in education, processes, ideas)".
(Waight, 2014)	Technology is delineated as something that "improves and makes life easier, the artifacts which function to accomplish tasks, and the representations of advances in civilization".
(Volti, 2005)	Technology is "a system created by humans that uses knowledge and organization to produce object and techniques for attainment of specific goals".
(Ramanathan, 1994)	Technology is the manifestation of four elementary and interacting components: technoware, related to the tangible and palpable parts (i.e. tools and systems);humanware, related to the human resources who, with their knowledge and skills, produce, use and express the technoware; orgaware, referred to effective organizational practices, linkages, and related arrangements needed to make the best use of technoware and humanware; and inforware, that represents the accumulation of knowledge by human beings related to the other 3 components.

Table 2.3: *Definition of technology*

CHAPTER 3

Concentration of Linguistic Knowledge Within BERT Embeddings

This Chapter focuses on studying the degree and localization of linguistic knowledge embedded within the representations of text created by Language Models, particularly BERT, part of the content of this Chapter has been used in the writing of Puccetti et al. (2021b).

With the goal of understanding how knowledge is arranged withing BERT representations of language, I define two research questions, aimed at: (i) investigating the relationship between the sentence-level linguistic knowledge encoded in a pre-trained version of BERT and the number of individual units involved in the encoding of such knowledge; (ii) understanding how these sentence-level properties are organized within the internal representations of BERT, identifying groups of units more relevant for specific linguistic tasks.

I define a suite of probing tasks based on a variable selection approach, in order to identify which units in the internal representations of BERT are involved in the encoding of similar linguistic properties. Specifically, I rely on a wide range of linguistic tasks, which can successfully model different typologies of sentence complexity (Brunato et al., 2020), from very simple features (such as sentence length) to more complex properties related to the morphosyntactic and syntactic structure of a sentence (such as the distribution of specific dependency relations).

3.1 Experimental Setup

To study how the information used by BERT to implicitly encode linguistic properties is arranged within its internal representations, I rely on a variable selection approach based on Lasso regression Tibshirani (1996), which aims at keeping as few non-zero coefficients as possible when solving specific regression tasks.

My goal is to identify which weights within sentence-level BERT internal representations can be set to zero, in order to understand the relationship between hidden units and linguistic competence and whether the information needed to perform similar linguistic tasks is encoded in similar positions.

I rely on a suite of 68 sentence-level probing tasks, each of which corresponds to a specific linguistic feature capturing characteristics of a sentence at different levels of granularity. In particular, I fit a Lasso

Level of Annotation	Linguistic Feature	Label
	Raw Text Properties	
Raw Text	Sentence Length	sent_length
	Word Length	char_per_tok
	Vocabulary Richness	
Vocabulary	Type/Token Ratio for words and lemmas	ttr_form, ttr_lemma
	Morphosyntactic information	
	Distibution of UD and language–specific POS	upos_dist_*, xpos_dist_*
POS tagging	Lexical density	lexical_density
	Inflectional morphology	
	Inflectional morphology of lexical verbs and auxiliaries	xpos_VB-VBD-VBP-VBZ, aux_*
	Verbal Predicate Structure	
	Distribution of verbal heads and verbal roots	verbal_head_dist, verbal_root_perc
	Verb arity and distribution of verbs by arity	avg_verb_edges, verbal_arity_*
	Global and Local Parsed Tree Structures	
	Depth of the whole syntactic tree	parse_depth
	Average length of dependency links and of the longest link	avg_links_len, max_links_len
	Average length of prepositional chains and distribution by depth	avg_prep_chain_len, prep_dist_*
Dependency Parsing	Clause length	avg_token_per_clause
	Order of elements	
	Order of subject and object	subj_pre, obj_post
	Syntactic Relations	
	Distribution of dependency relations	dep_dist_*
	Use of Subordination	
	Distribution of subordinate and principal clauses	principal_prop_dist, subordinate_prop_dist
	Average length of subordination chains and distribution by depth	avg_subord_chain_len, subordinate_dist_1
	Relative order of subordinate clauses	subordinate_post

Table 3.1: *Linguistic Features used in the experiments.*

regression model that takes as input layer-wise BERT representations of each sentence of a gold standard Universal Dependencies (UD) Nivre et al. (2016) English dataset and predicts the actual value of a given sentence-level feature. Lasso regression consists in adding an L_1 penalization to the usual ordinary least square loss. To do so, one of the most relevant parameters is λ, which tunes how relevant the L_1 penalization is for the loss function.

I perform a grid search with cross validation for each feature-layer pair, in order to identify the best suited value for λ according to each task. Specifically, my goal is to find the most suited value for seeking the best performance while having as few non-zero coefficients as possible.

3.1.1 Model and Data

I use a pre-trained version of BERT (BERT-base uncased, 12 layers). In order to obtain the representations for the sentence-level tasks I experiment with the activation of the first input token (*[CLS]*) and the mean of all the word embeddings for each sentence (*Mean-pooling*).

With regard to the data used for the regression experiments, I rely on the Universal Dependencies (UD) English dataset. The dataset includes three UD English treebanks: UD_English-ParTUT, a conversion of a multilingual parallel treebank consisting of a variety of text genres, including talks, legal texts and Wikipedia articles Sanguinetti and Bosco (2015); the Universal Dependencies version annotation from the GUM corpus Zeldes (2017); the English Web Treebank (EWT), a gold standard universal dependencies corpus for English Silveira et al. (2014). Overall, the final dataset consists of 23,943 sentences.

3.1.2 Linguistic Features

As mentioned above, I define a suite of probing tasks relying on a wide set of sentence-level linguistic features automatically extracted from the parsed sentences in the UD dataset. The set of features is based on the ones described by Brunato et al. (2020) which are acquired from raw, morpho-syntactic and syntactic levels of annotation and can be categorised in 9 groups corresponding to different linguistic phenomena. As shown in Table 3.1, these features model linguistic phenomena ranging from raw text one, to morpho-syntactic information and inflectional properties of verbs, to more complex aspects of sentence structure modeling global and local properties of the whole parsed tree and of specific subtrees, such as the order of subjects and objects with respect to the verb, the distribution of UD syntactic relations, also including features referring to the use of subordination and to the structure of verbal predicates.

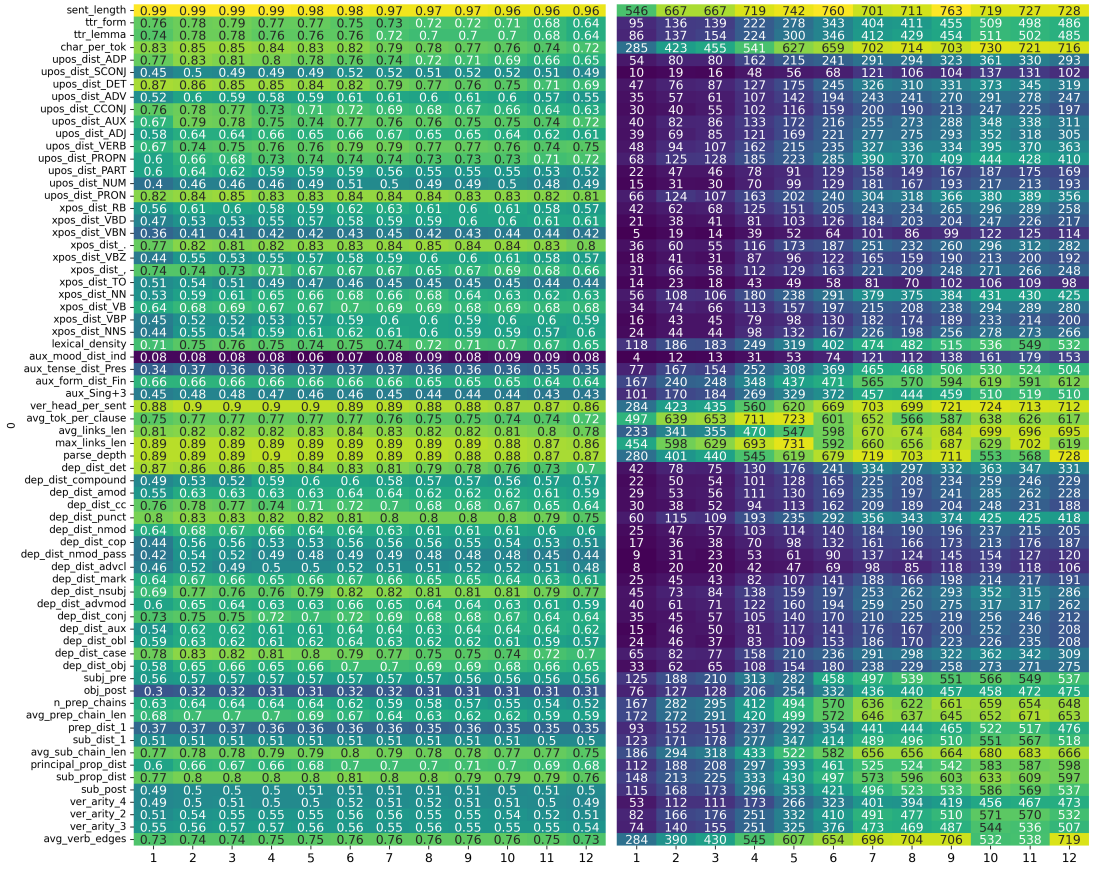


Figure 3.1: *Layerwise Spearman Correlation results for all the probing tasks (left heatmap) along with the number of non-zero coefficients (right heatmap) obtained with the sentence representations computed using the [CLS] token.*

3.2 Linguistic Competence and BERT Units

As a first analysis, I investigate the relationship between the implicit linguistic properties encoded in the internal representations of BERT and the number of individual units involved in the encoding of these properties. Figure 3.1 and Figure 3.2 report layer-wise Spearman correlation for all the probing tasks along with the number of non-zero coefficients obtained with the sentence representations computed with the *[CLS]* token and the *Mean-pooling* strategy respectively. As a first remark, one can notice that the *Mean-pooling* method proves to be the best one for almost all the probing features across the 12 layers. Moreover, in line with Hewitt and Manning (2019b), it is also noticeable that there is high variability among different tasks, whereas less variation occurs among the model layers.

One can observe that best scores are related to features belonging to raw text and vocabulary proprieties, such as sentence length and Type/Token Ratio. Nevertheless, BERT representations implicitly encode information also related to more complex syntactic features, such as the order of the subject (*subj_pre*) or the distribution of several dependency relations (e.g. *dep_dist_det*, *dep_dist_punct*).

Interestingly, the knowledge about POS differs when one considers more granular distinctions. For instance, within the broad categories of verbs and nouns, worse predictions were obtained by sub-specific classes of verbs based on tense, person and mood features (see especially past participle, *xpos_dist_VBN*). Similarly, within the verb predicate structure properties, I observe that lower Spearman correlations were obtained by features related to sub-categorization information about verbal predicates, such as the distribution of verbs by arity (*verbal_arity_**).

Focusing instead on the relationship between correlation and number of non-zero coefficients, it

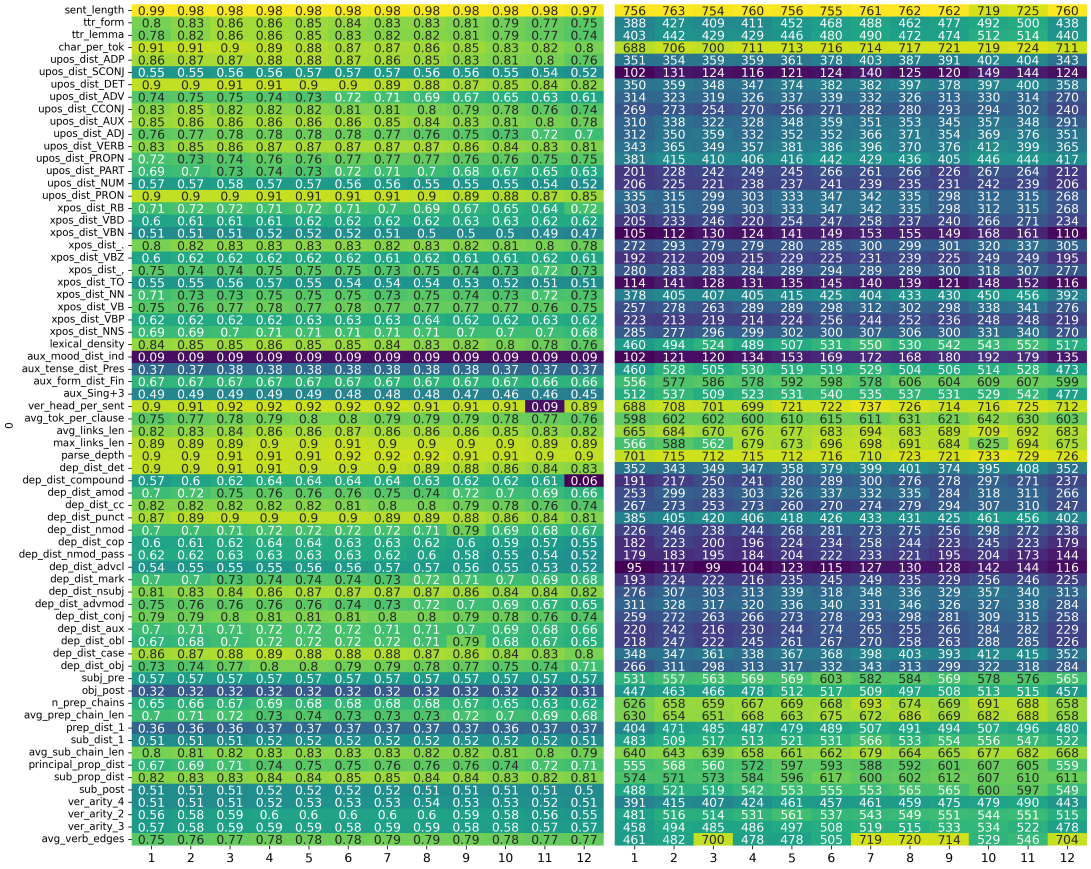


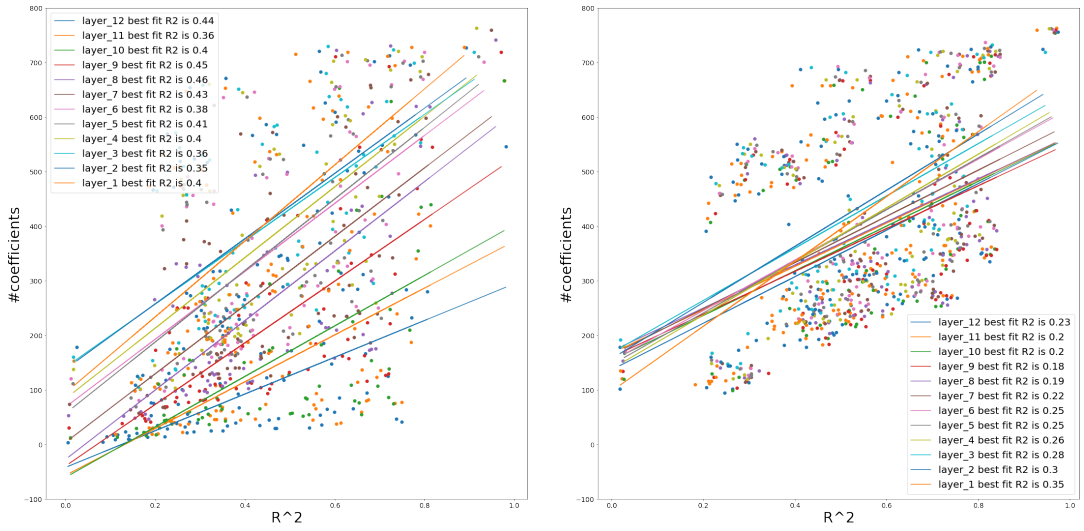
Figure 3.2: *Layerwise Spearman Correlation for all the probing tasks (left heatmap) along with the number of non-zero coefficients (right heatmap) obtained with the sentence representations computed with the Mean-pooling strategy.*

is noticeable that although best scores are achieved at lower layers (between layers 1 and 5 for both configurations), the highest number of non-zero coefficients occurs, instead, at layers closer to the output. This is particularly evident for the results achieved using the *[CLS]* token, for which there is a continuous increase across the 12 layers in the number of units used by the the probing models.

For both configurations, features more related to the structure of the whole syntactic tree are those for which less units were set to zero during regression (e.g. *max_links_len*, *parse_depth*, *n_prepositional_chains*), while properties belonging to word-based properties (i.e. features related to POS and dependency labels) were predicted relying on less units. Moreover, one can notice that features related to specific POS and dependency relationships are also those that gained less units through the 12 layers (e. g. *xpos_dist_.*, *xpos_dist_AUX*).

On the contrary, features belonging to the structure of the syntactic tree tend to acquire more non-zero units as the output layer is approached. This is particularly evident for the linguistic features predicted using sentence representations computed using the *[CLS]* token (e.g. *subj_pre*, *parse_depth*, *n_prepositional_chains*). I believe this is due to the interdependence between different units in each representation which tends to increase across layers, thus making the information less localized especially for those features that belong to the whole structure of the syntactic tree. This is coherent with the fact that using the *Mean-pooling* strategy a higher number of non-zero coefficients was preserved also in the very first input layers, suggesting that this strategy increases the interdependence between each unit and makes the extraction of localized information more complex.

In order to focus more closely on the relationship between probing results and non-zero units, Figure 3.3a and Figure 3.3b report the average R^2 scores versus average number of non-zero coefficients,



(a) **(b)**

Figure 3.3: *Average R^2 scores versus average number of non-zero coefficients, along with the line of best fit, for each layer and according to [CLS] (a) and Mean-pooling (b) strategy.*

along with the line of best fit, for each layer and according to the *[CLS]* token and to the *Mean-pooling* strategy respectively. Interestingly, for both *[CLS]* and *Mean-pooling* representations, R^2 scores tend to improve as the number of non-zero coefficients increases. Moreover, when considering sentence representations computed with the *[CLS]* token, this behaviour becomes more pronounced as the output layer is reached. This is in line with the fact that interdependence between different units tends to increase across layers, especially when taking into account representations extracted without using a mean-pooling strategy.

In order to investigate more in depth the behaviour of BERT hidden units when solving the probing tasks, I focus more closely on how the different units in the internal representations are kept and lost across subsequent layers. Figure 3.4 reports the average number of non-zero coefficients in a layer that are set to zero in the following one (Figure 3.4a), the average number of zero coefficients in a layer that are set to non-zero in the following one (Figure 3.4b) and the average value of the difference between the number of non-zero coefficients at pairs of consecutive layers (Figure 3.4c). As one can observe, there is high coherence between each layer and its subsequent one, meaning that the variation in the number of selected coefficient is stable (Figure 3.4c). However, the first two plots also show that there is a higher variation when considering non-zero coefficients in the same positions between pairs of layers. This underlines the fact that the information is not localized within BERT's internal representations, since the algorithm shows a degree of freedom in which units can be zeroed and which cannot.

In Figure 3.5 I report how many times each individual unit in the *[CLS]* (Figure 3.5a) and *Mean-pooling* (Figure 3.5b) internal representations has been kept non-zero when solving the 68 probing tasks for all the 12 BERT layers (816 regression task). In general, one can observe that the regression tasks performed using sentence-level representations obtained with the *Mean-pooling* strategy tend to use more hidden units with respect to the *[CLS]* ones. It is also interesting to notice that there is a highly irregular unit (number 308) that has been kept different from zero in a number of tasks and layers much higher than the average. This could suggest that this unit is particularly relevant for encoding almost all the linguistic properties devised in these probing tasks.

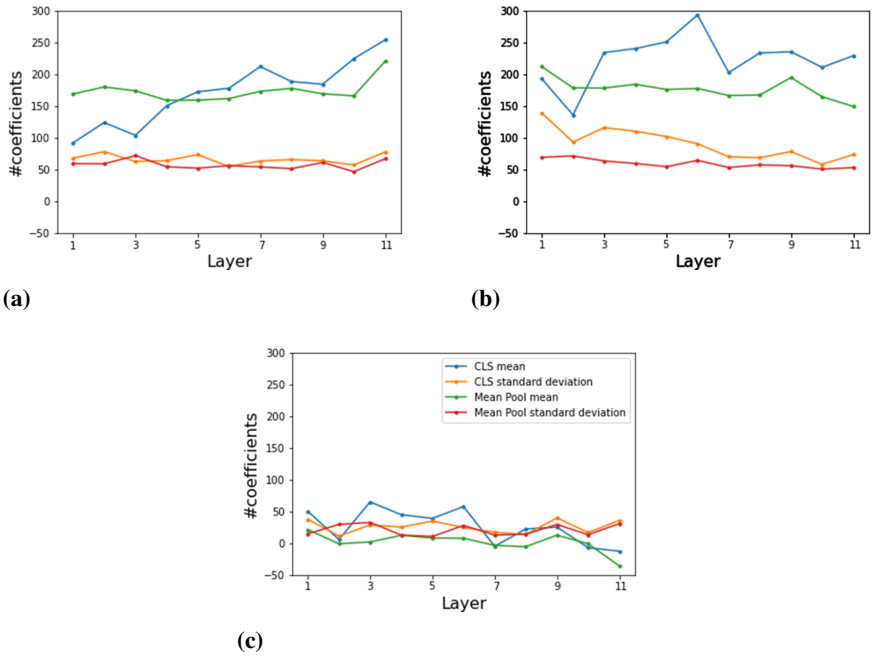


Figure 3.4: *In (a) the average number of non-zero coefficients in a layer that are set to zero in the following one (average number of dropped coefficients), in (b) the average number of zero coefficients in a layer that are set to non-zero in the following one (average number of gained coefficients) and in (c) the value of the difference between the number of non-zero coefficients at pairs of consecutive layers (average number of changed coefficients).*

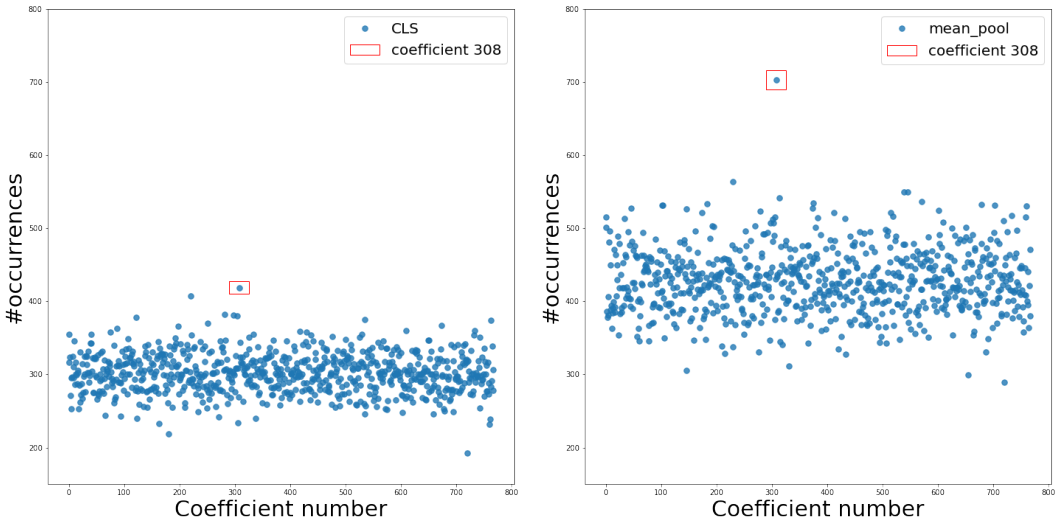
3.3 Linguistic Information Within BERT Representations

After having investigated the relationship between the linguistic knowledge implicitly encoded by BERT and the number of individual units involved in it, I verified whether one can identify groups of units particularly relevant for specific probing tasks. To this end, I clustered the 68 probing features according to the weights assigned by the regression models to each BERT hidden unit. Specifically, I perform hierarchical clustering using correlation distance as distance metric.

Figure 3.6 and Figure 3.7 report the hierarchical clustering obtained with the *[CLS]* and *Mean-pooling* internal representations at layers 1, 5 and 12. I choose layers 1 and 12 in order to study differences among the clustering of linguistic features taking into account the representations that were more distant and closer to the language modeling task respectively, while layer 5 is chosen since it is the layer after which BERT's representations tend to lose their precision in encoding this set of linguistic properties.

As a general remark, one can notice that, despite some variations, the linguistic features are organized in a similar manner across the three layers and for both the configurations. This is to say that, despite the number of non-zero coefficients varies significantly between layers and according to the strategy for extracting the internal representations, the way in which linguistic properties are arranged within BERT embeddings is consistent. This suggests that there is a coherent organization of linguistic features according to non-zero coefficients that is independent from the layer and the aggregation techniques taken into account.

Focusing on specific groups of features, one can see that, even if the traditional division with respect to the linguistic annotation levels (see Table 3.1) has not been completely maintained, it is possible to identify different clusters of features referable to the same linguistic phenomena for all the 3 layers taken into account and for both configurations. In particular, one can clearly observe groups of features related to the length of dependency links and prepositional chains (e.g. *max_links_len*, *avg_links_len*, *n_prepositional_chains*), to vocabulary richness (*ttr_form*, *ttr_lemma*), to properties related to verbal



(a) **(b)**

Figure 3.5: *Number of times in which each BERT individual unit (computed with [CLS] token in (a) and with Mean-pooing aggregation strategy in (b)) has been kept as non-zero when solving all the probing tasks for all the 12 layers.*

predicate structure and inflectional morphology of auxiliaries (e.g. *xpos_dist_VBD*, *xpos_dist_VBN* *aux_form_dist_Fin*, *aux_tense_dist_pres*) and to the use of punctuation (*xpos_dist_.*, *xpos_dist_,*, *dep_dist_punct*) and subordination (e.g. *subordinate_dist_1*, *subordinate_post*). Interestingly enough, BERT representations also tend to put together features related to each other but not necessarily belonging to the same linguistic macro-category. This is the case, for instance, of characteristics corresponding to functional properties (e.g. *upos_dist_ADP*, *dep_dist_det*).

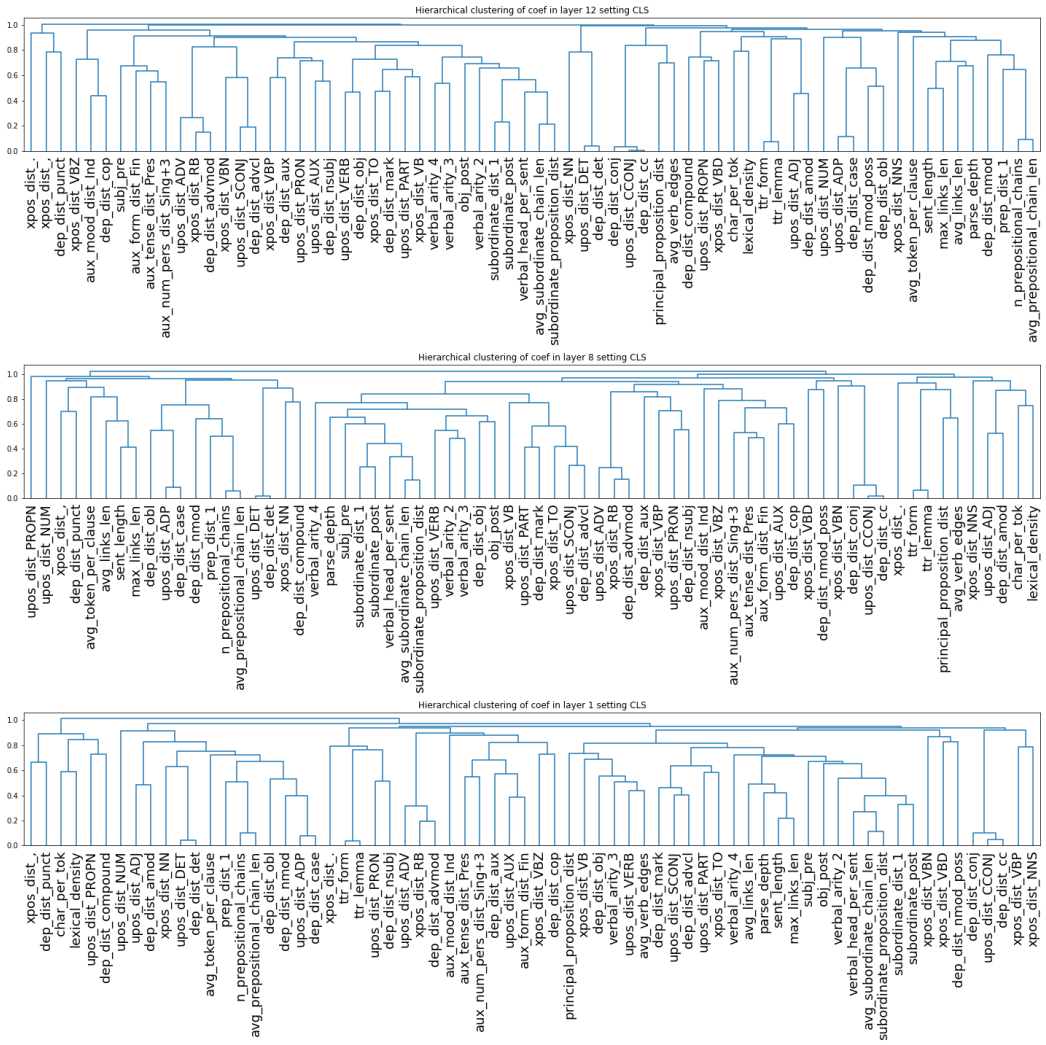


Figure 3.6: *From top to bottom, the hierarchical clustering for the* [CLS] *setting of all the tasks respectively at layers 1, 5 and 12.*

3.4 Conclusions

In this Chapter I propose an in-depth investigation aimed at understanding how BERT embeddings encode and organize linguistic competence. Relying on a variable selection approach applied on a suite of 68 probing tasks, I show the existence of a relationship between the implicit linguistic knowledge encoded by the NLM and the number of individual units involved in the encoding of this knowledge. I find that, according to the strategy for obtaining sentence-level representations, the amount of hidden units devised to encode linguistic properties varies differently across BERT layers: while the number of non-zero units used in the *Mean-pooling* strategy remains more or less constant across layers, the *[CLS]* representations show a continuous increase in the number of used coefficients. Moreover, I notice that this behaviour is particularly significant for linguistic properties related to the whole structure of the syntactic tree, while features belonging to part-of-speech and dependency tags tend to acquire less non-zero units across layers.

Finally, I find that it is possible to identify groups of units more relevant for specific linguistic tasks. In particular, I show that clustering sentence-level properties according to the weights assigned by the regression models to each BERT unit one can identify clusters of features referable to the same linguistic phenomena and this, despite some variations, is true for both the configurations and for all the BERT

layers.

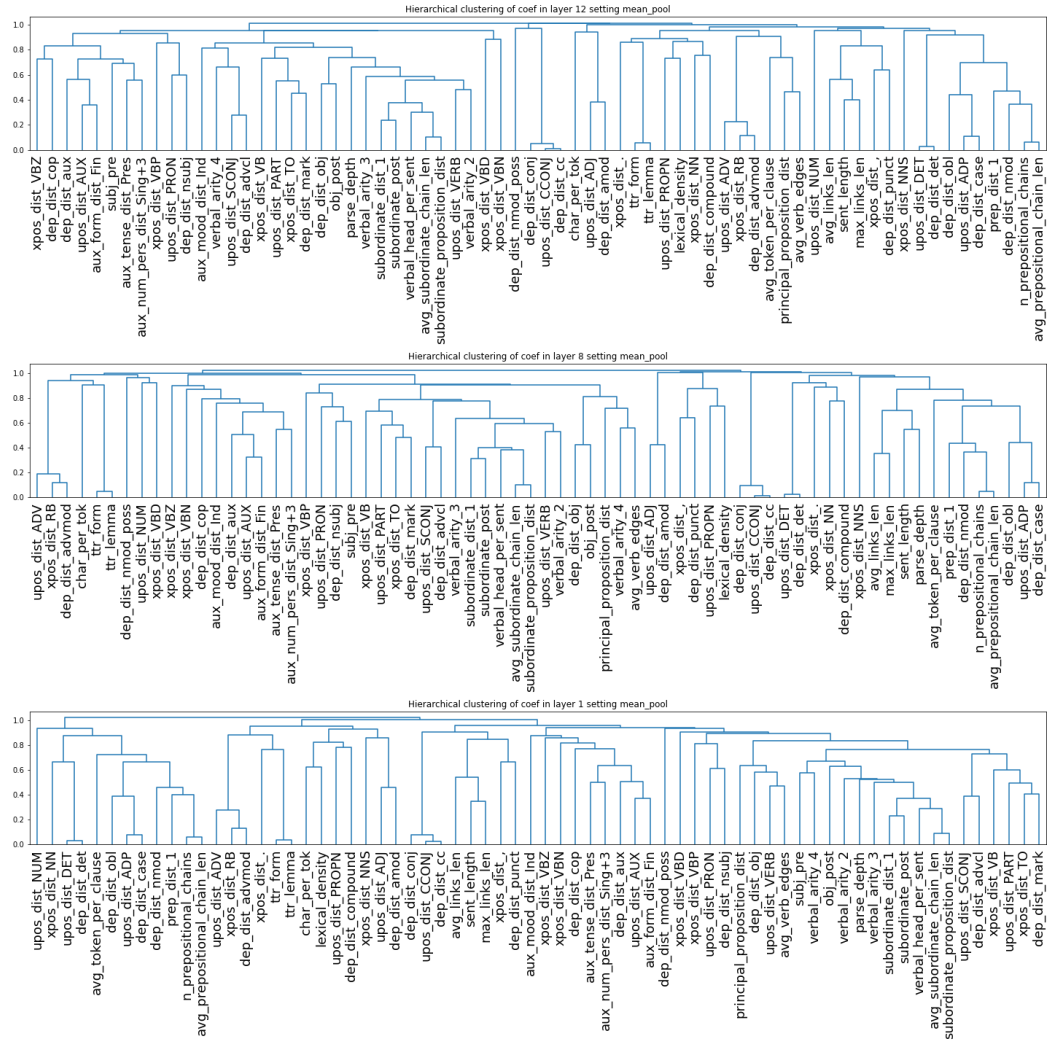


Figure 3.7: *From top to bottom, the hierarchical clustering for the* Mean-pooling *setting of all the tasks respectively at layers 1, 5 and 12.*

CHAPTER 4

The Impact of Token Frequency on Outlier Dimensions

This Chapter focuses on the study of *outliers*, weights of Language Models that show unusual behaviours. Throughout this Chapter I focus on studying BERT, while the Appendix of the thesis shows that similar results hold for RoBERTa. Part of the content of this Chapter has been used in the writing of Puccetti et al. (2022).

In Chapter 3 I show that there are few specific entries that are almost never excluded from BERT representations when performing feature selection.

This appears odd from a general perspective since current Transformer-based language models are heavily overparametrized, which explains why it is possible to prune these models by up to 30-40% (Gordon et al., 2020; Sanh et al., 2020; Prasanna et al., 2020; Chen et al., 2020, inter alia) without a significant drop in performance.

However, it has recently been shown that multiple Transformer-based language models (LMs) are highly sensitive to removal of *outlier dimensions* Kovaleva et al. (2021): the parameters (weights and biases) in the output element of a Transformer layer, the magnitude of which is unusually large within the layer (consistently in the same dimension across the model layers). For BERT model family the output element is the LayerNorm, as shown in Figure 4.1.

These parameters are the same that I have found to be relevant to the vast majority of linguistic probing tasks, furthermore, although these parameters constitute less than 0.0001% of the full BERT (Devlin et al., 2019) model, removing them significantly degrades BERT's performance. Puccetti et al. (2021b) find that the same parameters are particularly relevant in several linguistic probing tasks. These dimensions affect the vector representation of different tokens in the same way, making the embedding space less isotropic and thus reducing its representational power Liang et al. (2021). Outlier dimensions have also been found to make model quantization challenging Bondarenko et al. (2021); Dettmers et al. (2022) as they need to be treated separately from others when defining quantization schemes. Thus there are both conceptual and practical reasons supporting a deeper study of this phenomenon.

What is not clear at this point is the mechanism behind the emergence of outliers. I replicate the original findings in BERT and RoBERTa, and contribute new evidence **directly linking the outlier phenomenon with the frequency of encoded tokens in the pre-training data, as well as the self-attention pattern focusing on special tokens**. I also present evidence for two kinds of outliers: some

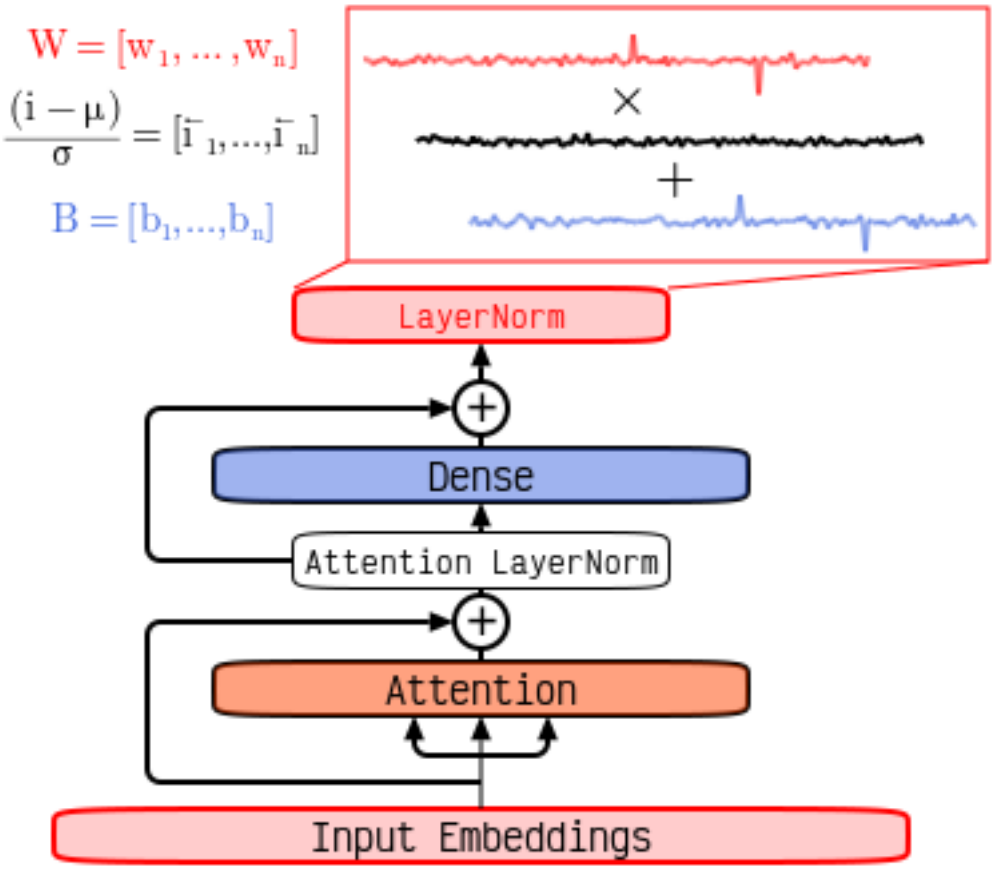


Figure 4.1: *The Transformer Layer architecture diagram with outliers at the normalization layer (LayerNorm).*

of them affect the Masked Language Model (MLM) performance the most in the middle layers (where the correlation with token frequency is at its peak), and for others the impact grows towards the final layers (although correlation with token frequency decreases). This work contributes to mechanistic understanding of Transformer-based LMs, and it might be useful for future research on decreasing anisotropy in pre-trained LMs.

4.1 Methodology and Experimental Setup

According to Kovaleva et al., *outliers* are parameters (both weights and biases) in the final element of a Transformer layer (LayerNorm for BERT family, final MLP for GPT-2), which have unusually high magnitude[1] within the layer. *Outlier dimensions* are those dimensions at which outlier parameters are found consistently across the model layers.

The reason Kovaleva et al. study these parameters is that when they are disabled, the model performance on downstream tasks is significantly reduced. Since not all parameters that can be identified by the magnitude and position criteria have that effect, I add this property to the definition. In this work, the term *outlier dimension* refers to the dimensions with parameters meeting the magnitude criteria across layers and having at least 5x more damaging effect on accuracy on a representative downstream task, for which I choose MNLI.

To disable the outliers, unless stated otherwise, I set to zero both the LayerNorm *weight* and *bias* parameters for all layers (24 parameters in total for one outlier dimension in BERT and RoBERTa-base)[2]. See Section A.1 for the full list of outliers identified for all models in this study.

I use the notation O to refer to specific LayerNorm outlier parameters (in BERT model family): e.g. $O381$ to indicate "an outlier with index 381". Since outlier indices are a constant for a given model, in this study I will also discuss *hidden state outlier dimensions*: the coefficient of the hidden-state with the same index as the outlier.

I experiment with BERT-base Devlin et al. (2019) (*"bert-base-uncased"*), RoBERTa-base Liu et al. (2019b) (*"roberta-base"*) and Vision Transformer Dosovitskiy et al. (2021) (*"google/vit-base-patch16-224-in21k"*) from the *transformers* library[3]. For the experiments on pre-training dynamics I rely on the checkpoints with seed 1 provided by Sellam et al. (2022)[4].

The details of hardware, implementation, and energy expenditure are outlined in Chapter A. I release the code to replicate my experiments[5].

4.2 Outliers Phenomenon in Transformers

I start by replicating Kovaleva et al.'s experiments identifying the outliers for BERT- and RoBERTa-base ($O308$ and $O381$, $O77$ and $O588$ respectively), and their effect on downstream task performance.

Table 4.1 shows the average performance and standard deviation of BERT-base over 5 fine-tuning runs for eight GLUE tasks[6]. Thus I successfully replicate the original experiment on model degradation after[7] removal of the outliers. Since the effect is consistent across GLUE tasks, I use MNLI as a representative downstream task in the remaining experiments. I also confirm that RoBERTa-base behaves similarly (see Section A.2).

[1] The original definition of outliers is not entirely formal, and needs to be further specified for particular models: the magnitude of the outliers was within 2 standard deviations from the mean for RoBERTa, and within 3 for BERT.

[2] Note that this is equivalent to zeroing out the outlier of the hidden state generated by that layer.

[3] https://github.com/huggingface/transformers

[4] https://github.com/google-research/language/tree/master/language/multiberts

[5] https://github.com/gpucce/outliersvsfreq/tree/main

[6] I consider 8 GLUE Wang et al. (2018) tasks: CoLA Warstadt et al. (2019), SST Socher et al. (2013a), MRPC Dolan and Brockett (2005a), STSB Cer et al. (2017), MNLI Williams et al. (2018), QNLI Rajpurkar et al. (2016a) and RTE Bentivogli et al. (2009). I exclude WNLI task, which BERT is unable to "learn" Prasanna et al. (2020).

[7] Kovaleva et al. (2021) also show that if the outliers are removed *before* fine-tuning, the model is able to recover without any negative effects.

bert-base-uncased	cola	mnli	mnli-mm	mrpc	qnli	qqp	rte	sst2	stsb
baseline	56.9 +/- 1.5	84.5 +/- 0.2	84.8 +/- 0.4	84.3 +/- 1.1	91.4 +/- 0.1	91.1 +/- 0.1	66.3 +/- 1.6	92.8 +/- 0.5	89.0 +/- 0.3
1 random removed	56.5 +/- 1.5	84.5 +/- 0.2	84.8 +/- 0.4	84.5 +/- 0.9	91.3 +/- 0.1	91.1 +/- 0.1	66.6 +/- 1.7	92.8 +/- 0.4	89.0 +/- 0.3
w/o 308	47.3 +/- 1.2	81.4 +/- 1.1	82.2 +/- 1.1	54.0 +/- 12.3	88.9 +/- 0.8	89.1 +/- 1.6	62.1 +/- 3.2	90.8 +/- 1.1	56.9 +/- 17.4
w/o 381	33.8 +/- 9.4	73.2 +/- 2.1	73.8 +/- 2.0	64.6 +/- 15.2	80.3 +/- 1.5	79.8 +/- 3.2	55.8 +/- 1.6	87.9 +/- 1.0	78.1 +/- 4.8
2 random removed	56.4 +/- 1.5	84.5 +/- 0.2	84.8 +/- 0.4	84.3 +/- 0.9	91.3 +/- 0.1	91.1 +/- 0.1	66.6 +/- 1.7	92.8 +/- 0.5	88.9 +/- 0.3
w/o 308 & 381	15.9 +/- 4.2	58.4 +/- 3.3	59.0 +/- 3.5	55.1 +/- 16.3	74.5 +/- 1.3	74.6 +/- 4.6	55.3 +/- 4.5	76.0 +/- 2.4	35.7 +/- 15.3

Table 4.1: *Average BERT scores over 5 runs on GLUE benchmarks with the effect of outlier removal. The rows* 1 random removed *and* 2 random removed *show the average over 5 removals of random non outliers (1 or 2 at a time respectively) for 5 different fine-tuned models.*

Outliers removed	CIFAR10	CIFAR100
Full model	98.6	92.5
1 random dimension	98.6	92.5
$O759$	98.6	92.3
$O187$	98.6	90.5
2 random dimensions	98.6	92.4
$O759$ + $O187$	98.5	**84.9**

Table 4.2: *Outlier removal effect for Visual Transfromer.*

4.2.1 Outliers in Other Transformers

Kovaleva et al. (2021) focus exclusively on Transformer-based LMs. To establish whether outliers could be something specific to pre-training on language data, I investigate the presence of outliers in the Vision Transformer (ViT) Dosovitskiy et al. (2021). Table 4.2 shows ViT accuracy on CIFAR10 Krizhevsky et al. (2009) and CIFAR100 Krizhevsky (2009): image classification tasks with a choice between 10 and 100 possible labels respectively. Using the magnitude and position criteria I identify candidates $O759$ and $O187$ and experiment with disabling one or both of them, as well as randomly selected dimensions as a control. For this model, the accuracy on MNLI can't be used as a measure for outliers, instead I use the accuracy on CIFAR100.

I see that, for CIFAR100, with both outliers disabled the model experiences $\approx 7\%$ loss in accuracy, but that does not happen for CIFAR10. The reason for that could be that CIFAR10 is a much simpler task, on which the model achieves above 98.5% accuracy. If the model succeeds in positioning the small number of classes sufficiently far apart in the representation space, then even the loss of outliers might be insufficient to disrupt that. If that is the reason for the discrepancy between CIFAR10 and CIFAR100, then perhaps the 100-class classification is still an easier problem than the GLUE tasks, for which BERT degrades in performance significantly more (see Table A.2).

I also explored two other Transformer-based models: ESM trained on protein sequences Rao et al. (2021) and Wav2Vec trained on audio data Baevski et al. (2020). I found no evidence for outliers there. This could be due to the fact that both of these models have a very small "vocabulary" (30-40 "tokens" vs tens of thousands for LMs).

4.2.2 Emergence of Outliers in Pre-Training

Kovaleva et al. (2021) pre-train a BERT-medium model for up to 250,000 steps. They find that outliers emerge relatively early in pre-training (step 50,000), and at about the same time LM perplexity starts to

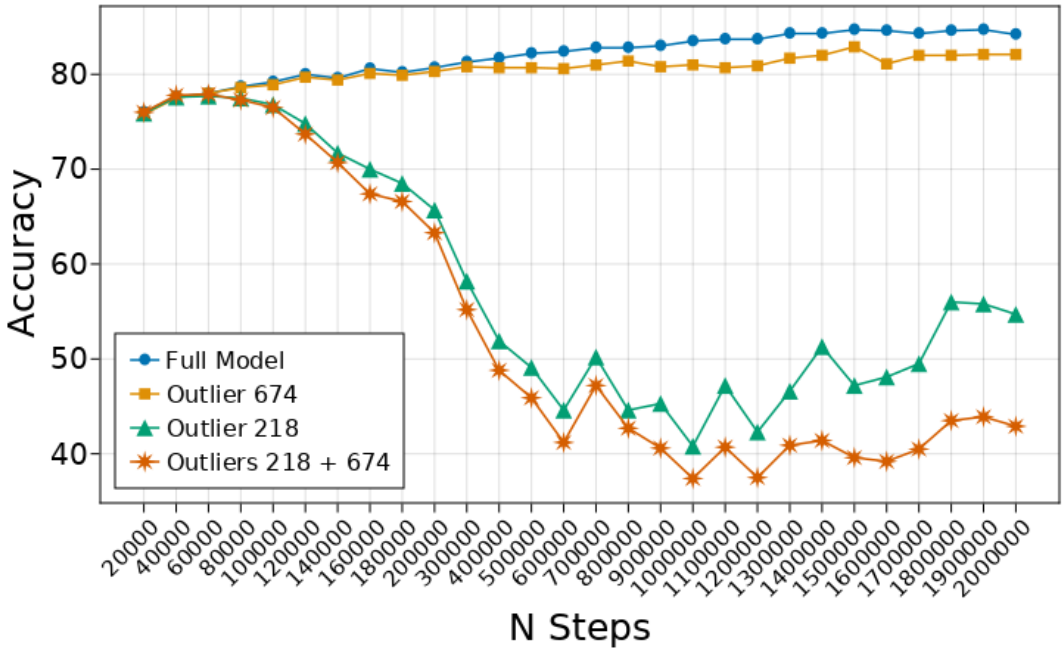


Figure 4.2: *The accuracy on MNLI-matched of the checkpoints for BERT-base (seed 1) by Sellam et al. (2022) for full model or with each outlier removed.*

improve. A limitation of this experiment is a relatively small model and the fact that both observed events coincide with the warm-up ending.

I examine the full BERT-base checkpoints released by Sellam et al. (2022), who pre-train five models from scratch with different random initialization. For each model, they release the checkpoints for every 20,000 steps between 0 and 200,000 steps, and after that – for every 100,000 steps up to 2,000,000. I use the seed numbered as 1 (zero indexed). Like BERT-base and RoBERTa-base (Section 4.1) and find that this BERT also has two outliers, $O218$ and $O674$, the same for all the checkpoints for this seed.

I investigate the main outlier effect: the drop in performance of the model fine-tuned on the chosen representative downstream task, MNLI-matched Williams et al. (2018). Figure 4.2 shows the accuracy for all the checkpoints from seed 1, comparing the full model with the model with $O218$, $O674$, and both $O218$ and $O674$ removed. The expected effect is clearly observed after step 80,000 for $O218$ and $O218$ + $O674$, but not $O674$ alone. This is consistent with the findings of Kovaleva et al. (2021) who also report various size of effects for outliers identified purely by magnitude. The results for MNLI-mismatched are similar and available in Section A.3.

After step 80,000 the full model steadily increases in accuracy, reaching 83.5% at step 10^6. Training for 10^6 more steps only achieves $\approx 1\%$ gain, illustrating the diminishing returns effect with further pre-training. The performance without outliers degrades over time, but at the later stages of pre-training (not observed by Kovaleva et al.) that trend is not steady: after $\approx 10^6$ steps the model accuracy with either $O218$ or $O218$ + $O674$ removed slowly grows over time, often with high variance between the "neighboring" checkpoints.

Another observation from Figure 4.2 is that after the first 10^6 steps[8] , the difference between the accuracy of the model without the most disrupting outlier $O218$ and $O674$ increases. This suggests that the dynamics for the two outliers are different: while one gains importance from the early stages of pre-training, the other one rises after more optimization steps[9]. This may be related to the different behavior

[8]Interestingly, the number of 10^6 steps is also the number of training steps mentioned in the original BERT paper Devlin et al. (2019), and even the models by Sellam et al. (2022) (also from Google) do not match the originally reported performance at the original amount of pre-training. Sellam et al. (2022) state that they need to train for twice as long to reach comparable performance on all the tasks from GLUE Wang et al. (2018) and SQuAD Rajpurkar et al. (2016c).

[9]Figure 4.2 shows the accuracy with different classification heads initialization. See Section A.3 for a similar case with fixed

for the hidden state dimensions corresponding to the two outliers, which I will present in Section 4.3.

4.3 The Impact of Outliers

4.3.1 Effects on Masked Language Modeling

So far it is known that disabling the outliers negatively affects BERT downstream task performance (Figure 4.2), but it is unclear *why* that happens. Since LMs rely on statistical patterns of token co-occurrence, token frequency in pre-training data[10] could be expected to affect the learned representations. I investigate whether outlier removal affects what kinds of tokens (in terms of their frequency in pre-training data) the MLM predicts.

Figure 4.3 shows the frequency of tokens predicted by the model over $200,000$ sentences from Wikipedia. I use the standard masking strategy: 15% tokens masked randomly. For BERT-base I observe that **the model with disabled outliers consistently predicts more tokens that were highly frequent in the training data**, and fewer tokens that were rare. RoBERTa shows a similar behavior (see Section A.2 for the details).

I also considered if the outliers impact the distribution of POS tags of the predicted tokens. I found that disabling $O381$ is the most disruptive and that, similarly to $O308$, it pushes the model towards predicting more nouns, punctuation, symbols, and adpositions (see Section A.4 for details).

4.3.2 Token Frequency Vs Performance

If outlier removal impacts the model ability to *predict* tokens it observed less often in pre-training (Subsection 4.3.1), could it also impact the model *encoding* of tokens more/less frequently seen in pre-training?

The LayerNorm outliers are an intrinsic property of the model itself. For this experiment I need to consider the interaction between the model and its input data. Hence I consider *the hidden state outlier dimensions*: the hidden state parameters at the dimensions corresponding to the outlier dimensions. They are the most affected by the outlier removal, since zeroing out a LayerNorm parameter removes precisely this component.

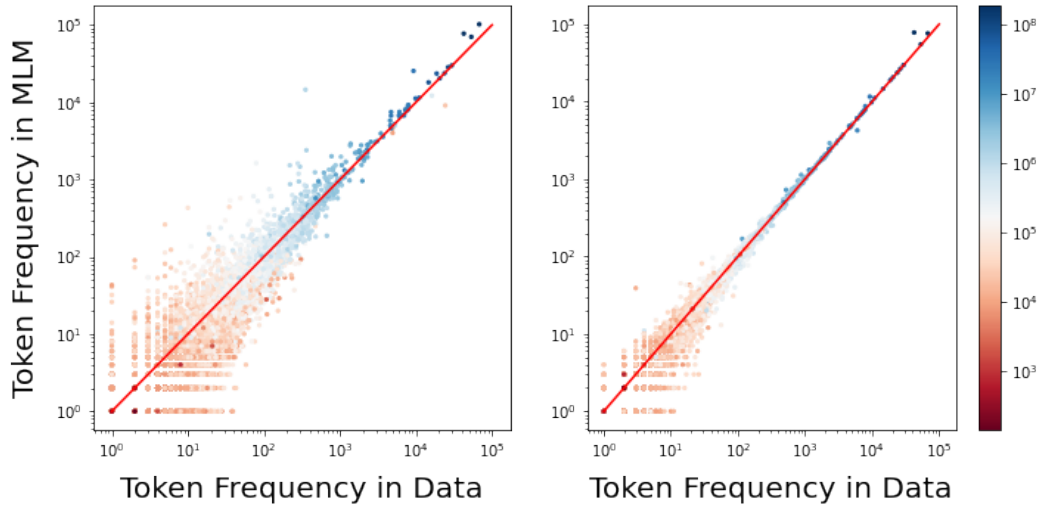
In this experiment I encode the validation set of Wikitext-v2 Merity et al. (2016) by BERT-base, and I measure the Pearson correlation between pre-training data frequency of encoded tokens, and the magnitude of the hidden state parameters corresponding to the outlier dimensions ($O308$ and $O381$) in each layer (see Chapter A for more details). The results are presented in Figure 4.4. I also track across all layers the main outlier effect (performance degradation when the outliers are disabled) in MLM and MNLI tasks, as shown in Figure 4.5.

I find that for the hidden state parameters corresponding to both $O308$ and $O381$ the correlation between their magnitude and encoded token frequency is much higher than for random dimensions, but they exhibit different layer-wise trends for that correlation vs impact on model performance:

Case 1: the magnitude of the hidden state parameters corresponding to the outlier dimensions is directly proportional to both its correlation with the encoded token frequency, and performance drop after removal of LayerNorm outlier parameters. For the hidden state dimension corresponding to $O381$, the correlation of the hidden state parameter magnitude with the encoded token frequency is closer to zero at the initial and final layers, and high in the middle layers (this trend continues until layer 9 when special tokens are included). Figure 4.5b shows that the removal of $O381$ has largest impact (in both MNLI and MLM) in layers 4-6. Coincidentally, Figure 4.4b shows that layers 4-6 are also the layers where the magnitude of hidden state dimension corresponding to $O381$ correlates with token frequency the most.

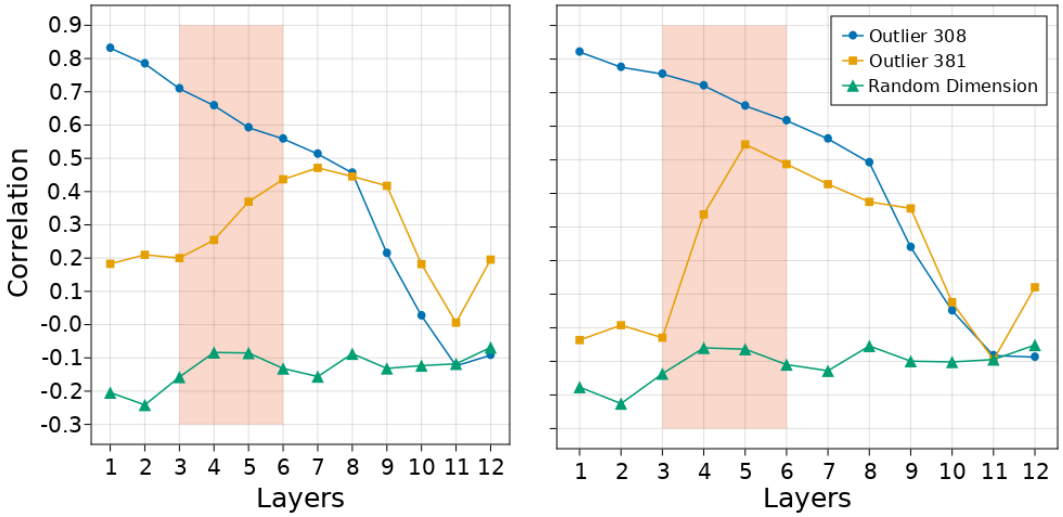
initialization.

[10]To estimate the frequency in the pre-training data I use a corpus similar to BERT pre-training data: it contains the Book Corpus Zhu et al. (2015) and Wikipedia dump from November 1st 2021.



(a) *Without Outliers* **(b)** *Full Model*

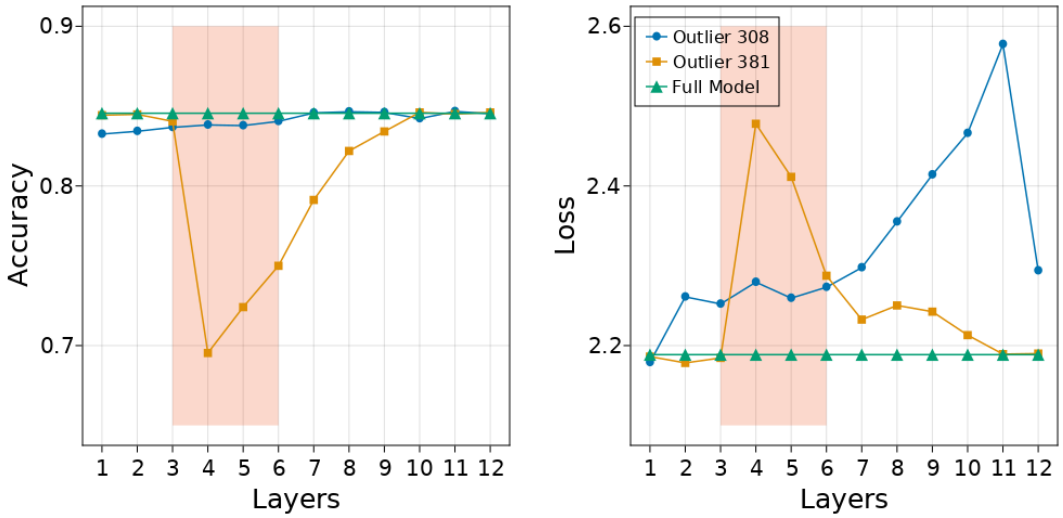
Figure 4.3: *A log-log scatter plot of token generation frequency vs true token frequency in data in MLM. The x-axis represents the number of time a token has been masked and the y-axis the times it has been predicted. The color shows the token appearances in pre-training data. In (a) for the* bert-base-uncased *model with zeroed out outliers and in (b) for the full pre-trained model.*



(a) *With special tokens* **(b)** *Without special tokens*

Figure 4.4: *BERT-base encoding Wikitext-v2 validation set data: the correlation between magnitude of hidden state parameters corresponding to outlier dimensions, and frequency of encoded tokens in pre-training data.*

Case 2: the magnitude of the hidden state parameters corresponding to the outlier dimensions, and their correlation with the encoded token frequency are both inversely proportional to the performance drop after removal of LayerNorm outlier parameters. For $O308$ the pattern is the opposite: the magnitude of its corresponding hidden state parameter strongly correlates with encoded token frequency at the initial layers, but not in the final ones. However, Figure 4.5b shows that the removal



(a) *MNLI-m performance* **(b)** *MLM loss (in wikitext-v2)*

Figure 4.5: *BERT-base: effect of disabling outliers.*

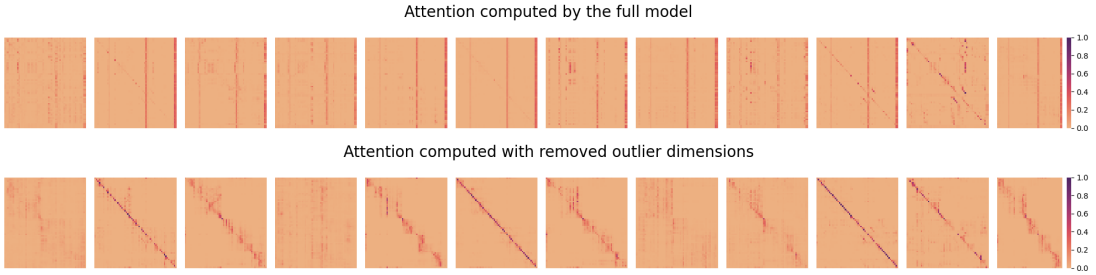


Figure 4.6: *The self-attention patterns at the 10th layer of the full 'bert-base-uncased' pre-trained model vs the same model with removed LayerNorm outliers.*
Encoded example from MNLI: [CLS] Thebes held onto power until the 12th Dynasty, when its first king, Amenemhet I who reigned between 1980 1951 b.c. established a capital near Memphis.[SEP] The capital near Memphis lasted only half a century before its inhabitants abandoned it for the next capital. [SEP]

of this LayerNorm outlier has a larger impact on MLM loss on the final layers[11]. As a result, the removal of $O308$ is less harmful for most downstream tasks as shown in Table 4.1 because fine-tuning mostly affects the layers closer to the output Liu et al. (2019a); Kovaleva et al. (2019), therefore it cancels a part of the effect of disabling this parameter.

To confirm that this is not a pattern specific to BERT I also perform the same experiments for RoBERTa-base, and I find that it also has the two kinds of outliers with the direct and inverse relationship to performance drop ($O588$ and $O77$ respectively). The data for these experiments is available in Section A.2.

Since BERT encodes sequences always starting with '[CLS]' and ending with '[SEP]', these special tokens could store positional information, and they are also highly frequent. Therefore I repeat the experiment discarding them (Figure 4.4b), but the overall trend is not affected.

[11]The main discrepancy in this pattern is the frequency correlation of the hidden state dimension corresponding to $O308$, and its MLM loss at the last layer. However, the lower loss can be a consequence of the parameter not affecting any following Transformer layer.

4.3.3 Outliers Effect on Attention

In Subsection 4.3.2 I show that there is a correlation between the magnitude of the hidden state parameters corresponding to outlier dimensions, and the token frequency in the pre-training data. Prior work Clark et al. (2019b); Kovaleva et al. (2019) showed that BERT self-attention often "points" to highly frequent tokens, including the special tokens and punctuation marks. Given this, my next question is whether the outliers also affect the self-attention patterns. As argued by Dong et al. (2021), attention alone would map tokens to very low dimensional spaces, and in that case the outlier phenomenon would be consistent with such a mapping.

I find that there is indeed such an effect. To illustrate it I encode a MNLI sample with BERT-base. Figure 4.6 shows the self-attention maps for the 12 heads of the 10th layer[12], directly comparing the self-attention in a full model vs a model with the outlier dimensions removed.

The most conspicuous difference is the fact that the vertical bars in the self-attention maps of the full model vanish once the outliers are zeroed out. This "vertical" attention pattern has been reported before Kovaleva et al. (2019), and in BERT it often corresponds to attention to special tokens and punctuation. It may seem that without the outliers the diagonal patterns become more salient, but in fact they are also present with the intact outliers, and their increased saliency in the plot is simply an effect of softmax normalization.

Figure 4.6 only shows a single example. To establish whether this effect is stable, I encode 1500 sequences from Wikitext-v2 validation set and measure the Pearson correlation between *average vertical attention value* of each token (the average over attention columns in the encoded sequence), and the magnitude of the hidden state parameters corresponding to the outlier dimensions. In cases of the "vertical" self-attention pattern, the average vertical attention value would be relatively high.

Figure 4.7 shows the results of this experiment, which I repeat with and without BERT special tokens ('[CLS]' and '[SEP]'). As a control, Figure 4.7c and Figure 4.7f show the average correlation over a sample of hidden state parameters at random dimensions. For the randomly picked weights the correlation is ≈ 0, which is expected (since these vectors have length 768, the individual dimensions of randomly sampled vectors should have a negligible contribution).

Compared to random dimensions, the hidden state parameters at dimensions corresponding to both $O308$ and $O381$ have on average a significantly higher correlation between their magnitude and average self-attention query values. This confirms that the pattern shown in Figure 4.6 is prevalent, and the tokens with high hidden state outlier dimension value tend to also have high average value over attention columns, i.e. **they are attended to by most other tokens**.

An unexpected pattern is represented by the negative correlations in Figure 4.7a and Figure 4.7d at initial and final layers. I argue that at early layers this happens because the vertical patterns are less frequent, while at the final ones because the outliers in those layers are less relevant. The trend is similar to what I observed in Figure 4.4.

I also observe several trends that mirror the observations from Subsection 4.3.2:

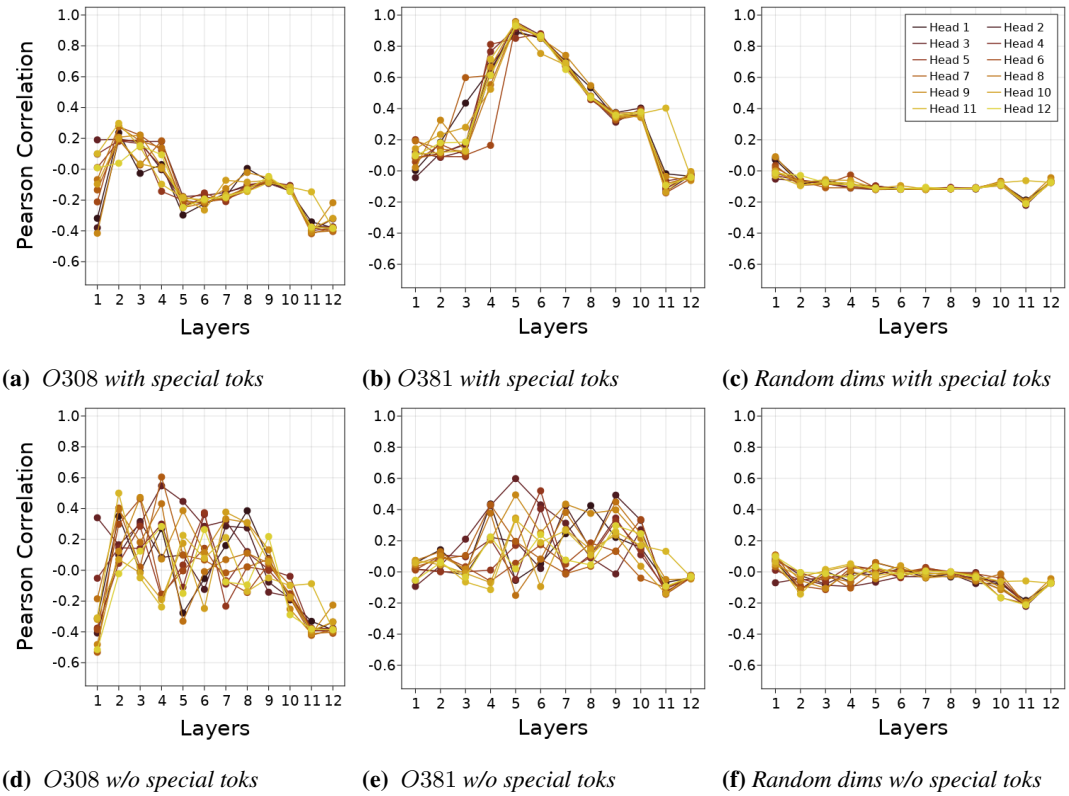
- The hidden state parameter value corresponding to $O308$ has a higher correlation with average vertical attention value since the initial layers (except the very first) which decreases at the final layers. For parameters corresponding to $O381$, the correlation grows at layer 4-5 and then vanishes at the final one. Both of these trends are consistent with Figure 4.4 showing the correlation to frequency.

- Special tokens affect these trends: Figure 4.7d and Figure 4.7e show that excluding them does not fundamentally change the pattern, but the results become less stable across heads.

- Both Figure 4.4 and Figure 4.7 show large variation as the information flows through the model, which suggests that the effect is not entirely formed at the model input.

Overall the results of this experiment suggest that **the relationship between the outlier phenomenon and encoded token frequencies in pre-training data also affects the self-attention mechanism of**

[12] I choose the 10th layer because prior work suggests that the layers closer to the output are more affected by fine-tuning and change their patterns more radically Kovaleva et al. (2019), and also encode more task-specific information Liu et al. (2019a).



(a) *O*308 *with special toks* **(b)** *O*381 *with special toks* **(c)** *Random dims with special toks*

(d) *O*308 *w/o special toks* **(e)** *O*381 *w/o special toks* **(f)** *Random dims w/o special toks*

Figure 4.7: *Each figure shows the correlation between the* average query values *in BERT-base self-attention heads, and the magnitude of hidden state parameters at the dimensions corresponding to outlier dimensions. The correlation is computed over examples from Wikitext-v2. Figures (c) and (f) show the average over 10 random dimensions.*

BERT. In particular, it affects the "vertical" attention pattern in which a token is attended to by most other tokens, and which was previously reported for the high-frequency special tokens. I confirm that RoBERTa self-attention exhibits a similar pattern (see Section A.2).

4.3.4 Exploring the Causes of Outliers

I have identified a correlation between the magnitude of hidden state parameters corresponding to outlier dimensions, and the frequency of the encoded tokens in the pre-training data. However, it is unclear whether the relationship is causal.

To establish that, I pre-train[13] from scratch 3 versions of BERT-medium as defined by Turc et al. (2019), with the following tokenization schemes:

- *SENTENCE*: I split sentences using a Spacy sentencizer[14], and add a '[SEP]' token at the end of each sentence and at the end of each encoded sequence. This is similar to the "full sentences" tokenization used to train RoBERTa Liu et al. (2019b).
- *CHUNK*: I add a single '[SEP]' token at the end of each sequence of 256 tokens rather than each sentence. The main effect that I expect from this is that the amount of '[SEP]' tokens is reduced roughly by a factor of 10.

[13]Except for the tokenization strategy, the training for each model is similar to the original BERT Devlin et al. (2019) with two exceptions: (a) the Wikipedia corpus is a more recent version, from 01/03/2022, (b) the max sequence length is 256 (instead of 512) and batch size 128 instead of 256 due to computational constraints (these appear to have limited effect on the MNLI benchmark). All models were trained for 327,500 steps.

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

	SPLIT	ONE SEP	RANDOMIZE
Full Model	79.6	79.2	76.8
Minus $O378$	**66.9**	**47.4**	-
Minus $O281$	78.8	-	-
Minus $O353$	-	-	75.8
Minus $O362$	-	-	**74.4**

Table 4.3: *Accuracy on MNLI-matched for each pre-training setting of BERT-medium.*

- *SENTENCE_FREQ*: Each sentence is followed by the '[SEP]' token, but I replace 50% of occurrences of regular tokens with frequency above 1.e-5 in the training corpus with a random token with a frequency below 1.e-5 in 50% of their occurrences[15].

Note that the RoBERTa-like *SENTENCE* tokenization is different from the classic BERT approach, where each encoded sequence always contains exactly 2 '[SEP]' tokens. The RoBERTa approach would make this token more frequent for the sequences containing more than one sentence, and hence also more appropriate for testing the frequency hypothesis.

Both *CHUNK* and *SENTENCE_FREQ* conditions corrupt the linguistic structure of the input, and so the trained MLM quality could be expected to drop as it acquires worse knowledge. But this setting will let me (a) identify the impact of token frequency on the outlier phenomenon, (b) disentangle the effect between frequent tokens in general and the '[SEP]' token.

All three models started from the same initialization but were fed different data according to the tokenization schemes. I find that *SENTENCE* model developed outliers $O281$ and $O378$, whereas in *CHUNK* the detrimental effect is only clear for $O378$. The *SENTENCE_FREQ* model developed two outliers: $O353$ and $O362$.

Table 4.3 shows all three BERT-medium models evaluated on MNLI-matched validation set as either the full model or with their respective outliers removed one by one. *SENTENCE* model is the best performing overall, but *CHUNK* is only .4 points behind the full model. Both of them develop a very damaging outlier $O378$, whereas the effect of $O281$ is less pronounced in *SENTENCE* and insignificant in *CHUNK*. Moreover, the single outlier in *CHUNK* is more damaging for the model. One possible explanation is that when the model has only one outlier, it likely relies on it more, which would result in higher performance degradation when it is disabled.

As expected, the *SENTENCE_FREQ* model that was fed the noisiest data performs worse than the other two (by $\approx 3\%$). But interestingly, it also does not develop any outliers as damaging as $O378$ is for the other two models.

I conclude that the frequency distribution of tokens in pre-training data contributes to the outlier phenomenon, and the '[SEP]' token is a part of that effect (since high frequency is one of the factors that characterizes it).

[15] Due to high computational costs of BERT pre-training I only experiment with one possible value of the threshold (1.e-5). When exactly tokens become "high frequency" for BERT-type MLMs remains a question for future work.

4.4 Discussion

4.4.1 Outliers in Transformer Pre-Training

Prior work Kovaleva et al. (2021) showed that the outliers are present in a large number of Transformer-based LMs. I provide complementary evidence for the Vision Transformer (Table 4.2). However, I was unable to identify outliers in protein and audio Transformers, which I attribute to significantly smaller vocabulary size. This finding hints towards the training data distribution being at the core of the outlier phenomenon.

Kovaleva et al. (2021) also show that outliers emerge early in pre-training (after 50K steps for BERT-medium). I extend that experiment by investigating the fully pre-trained BERT-base by Sellam et al. (2022), I find that the impact of outliers on the model grows up steadily until step 10^6. After that step the outlier effect is inconsistent between checkpoints, and the full model performance saturates. An interesting question for future work is what happens after outlier removal stops degrading model performance (around step 10^6), and whether it could be used as an early stopping criterion.

Transformer-based language models (LMs) have been shown to exhibit anisotropic behavior in their representations of both tokens and sentences Ethayarajh (2019); Gao et al. (2019); Rajaee and Pilehvar (2021); Timkey and van Schijndel (2021). While pervasive, this is an undesirable property because it reduces the average distance between tokens embeddings, and thus makes it more difficult to distinguish between tokens in the embedding space.

One consequence of the outliers growth over pre-training is that the attempts to remove anisotropy at the downstream task level Rajaee and Pilehvar (2021), although effective in some cases, could be only partially addressing the problem. In that case it might be more productive to change pre-training so as to better account for the skewed token frequency distribution Li et al. (2021b).

4.4.2 Outliers and Token Frequency

Li et al. (2021a) and Gao et al. (2019) show that embeddings of low frequency tokens lie further away from high frequency ones in the embedding space. In Subsection 4.3.2 I show how the outlier parameters influence the hidden state geometry proportionally to token frequency, and how this is more sensitive at earlier layers. This is consistent with findings of Li et al. (2021a) who show that the different geometry of frequent and non-frequent tokens is more evident for the layers closer to the input. Indeed, I observe this effect for $O381$ in BERT-base and $O588$ in RoBERTa-base (see Figure 4.4).

To the best of my knowledge, this is the first work to demonstrate the link not only between the geometry of the hidden states and frequency of encoded tokens in pre-training data, but also the model performance.

4.4.3 Outliers and Positional Embeddings

Concurrently with the demonstration of the outlier phenomenon by Kovaleva et al. (2021), Luo et al. (2021) attribute the high-magnitude weights to a different source: positional embeddings rather than LayerNorm weights. The positional embeddings could be expected to have more impact in the earlier layers. This thesis contributes to the dispute by showing that two different behaviours are present in both BERT-base and RoBERTa-base: one outlier dimension in the hidden states is disruptive in layers 4-6 ($O381$ for BERT and $O588$ for RoBERTa) while the other one at the layers 10-11 ($O308$ for BERT and $O77$ for RoBERTa).

The former also has a high correlation between the encoded token frequency and the magnitude of the hidden state parameters corresponding to outlier dimensions at initial layers, while the second has a low correlation at the final layers. This suggests that both mechanisms may play a role.

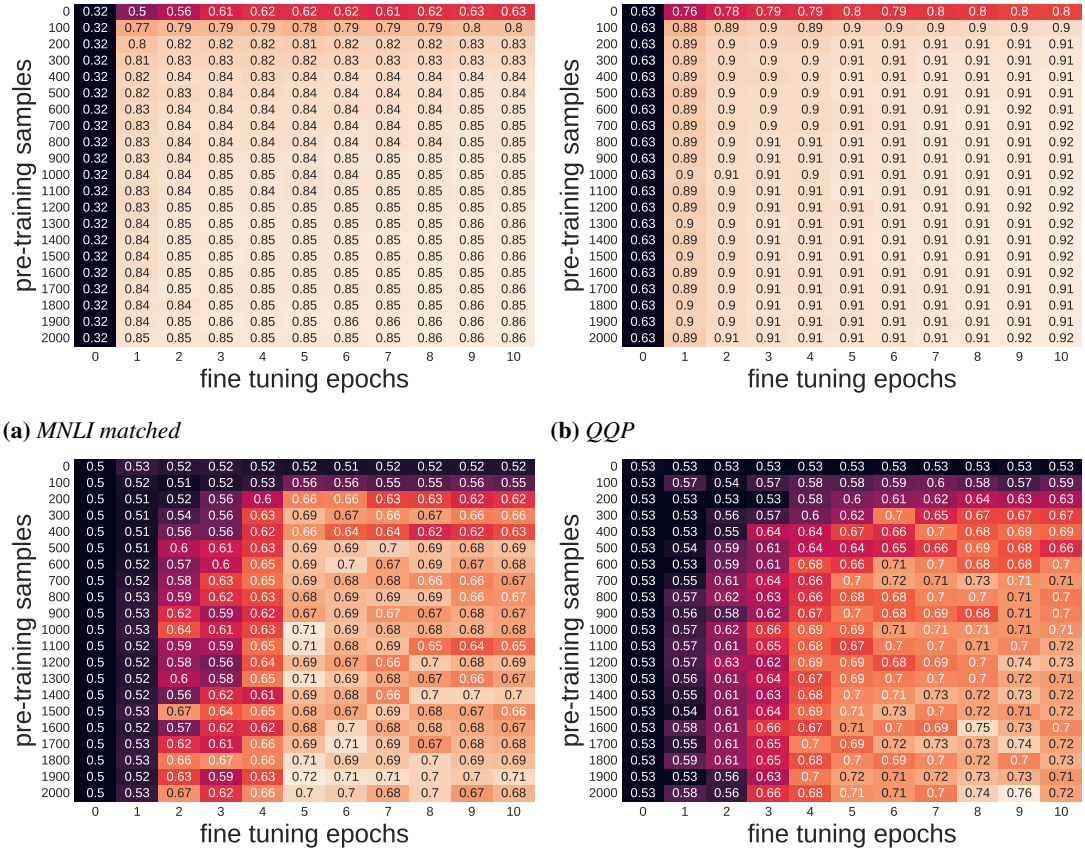
4.4.4 Outliers and Self-Attention

Kovaleva et al. (2019) identify 5 frequent self-attention patterns, 4 of which include vertical lines corresponding to special tokens. I show (Figure 4.7) that the presence of special tokens increases the correlation (in absolute value) between the average query value and the magnitude of the hidden state

dimensions corresponding to outlier dimensions. This suggests that the outlier phenomenon contributes to the vertical attention patterns identified by Kovaleva et al. (2019). From the computational perspective this is consistent with the attention being a bilinear form. Moreover, the relation between the outliers and the vertical self-attention pattern (often "pointing" to the highly frequent special tokens and punctuation) also hints at the relation between outliers and the token distribution in the pre-training data.

At the same time, the correlation remains evident in the final layers even when special tokens are ignored, indicating that the outliers also contribute to the attention shape more broadly. This is in line with Kobayashi et al. (2020) who argue that vertical patterns in attention do not indicate that no other information is encoded (hence simply norming the self-attention makes other relations more salient).

4.5 Pre-Training and Fine-Tuning Beyond Outliers



(a) *MNLI matched*

(b) *QQP*

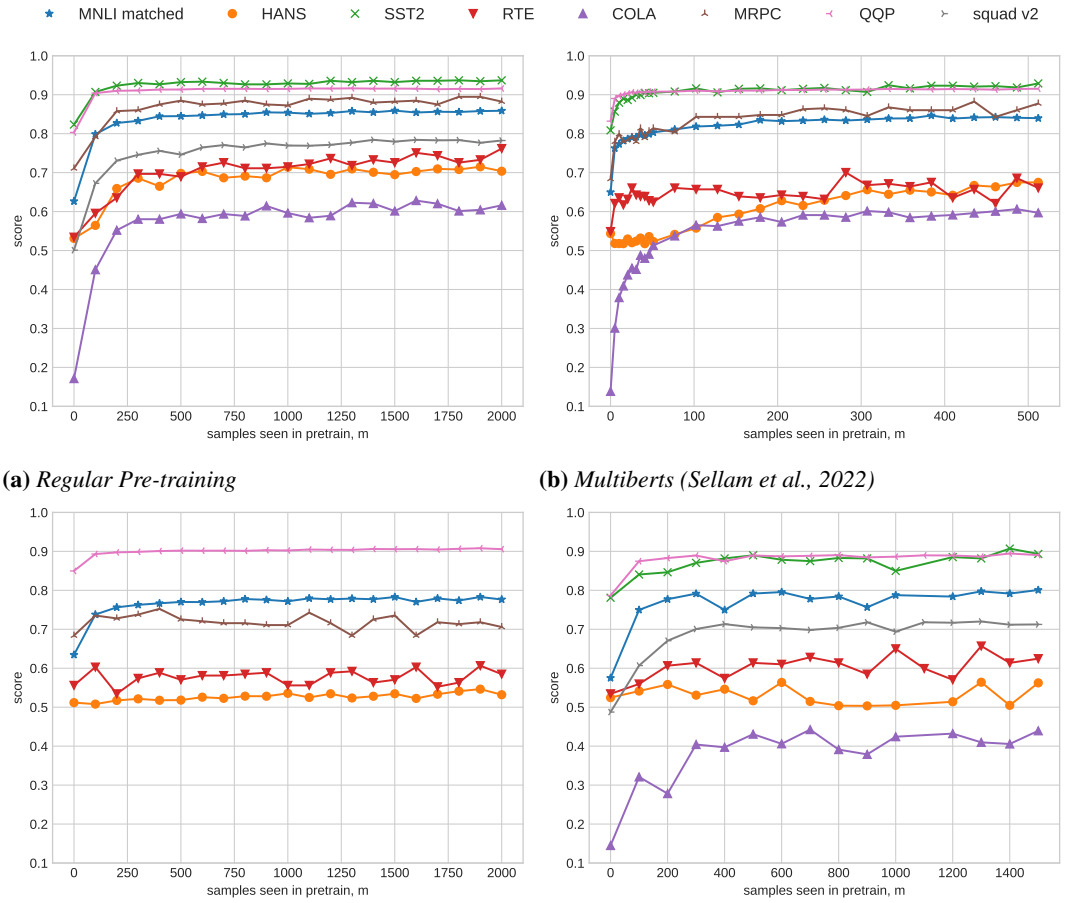
(c) *HANS*

(d) *RTE*

Figure 4.8: *Performance on each epoch of finetuning and each epoch of pretraining.*

After seeing how outliers are developed over a large share of the pre-training time of BERT like models (approximately half of the whole training) while knowing that after fine-tuning they harm performance on downstream tasks when removed, I investigate the relative importance of this two components of the training procedure.

This investigation stands between the fact that the more you pre-train the better downstream performance is (Liu et al., 2019b), and experiments suggesting that BERT can be fairly successfully fine-tuned on standard classification tasks starting from randomly initialized model Kovaleva et al. (2019) or even when pre-trained on non-linguistic tasks Kao and Lee (2021). I measure the relative importance of pre-training and fine-tuning in the success of LMs of the BERT family Devlin et al. (2019), using distinct architecture variants and widespread benchmarks for evaluation.



(a) *Regular Pre-training*

(b) *Multiberts (Sellam et al., 2022)*

(c) *Siamese model*

(d) *Shared model*

Figure 4.9: *Best downstream tasks validation performance on each stage of pre-training (a). Measuring on computationally comparable Multiberts checkpoints (Sellam et al., 2022) (b), with siamese architecture (c), same pre-trained model - Siamese architecture is constructed only during fine-tuning. Performance of an Albert (Lan et al., 2020b) equivalent version (d)*

To obtain a set of snapshots corresponding to different stages of pre-training, I use a language model pre-trained roughly following RoBERTa Liu et al. (2019b) methodology, together with a model with shared parameters, similar to ALBERT Lan et al. (2020a). This model uses same vocabulary and is trained on same data.

As a training corpus, I use a combination of a May 2022 English Wikipedia dump Foundation (6 01) (approximately 15GB of raw text) and a portion of C4 Raffel et al. (2020) corpus of equal size.

I perform aggressive filtering on source texts and reject any fragment with texts not detected with high confidence as English (using Python langdetect library), fragments with many non-roman characters, and some template passages which I find to be repeated many times through the corpus.

I set 256 tokens as sequence length, and use dynamic masking, based on independent rolls for each position (not fixing number of masked tokens per sequence) and use the same tokenization strategy as RoBERTa.

The training run has learning rate schedule as follows: linearly increase from 0 to 0.0096 for the first 6% of training, and then linearly decrease to 0. I use AdamW Loshchilov and Hutter (2019) optimizer with parameters $B_1 = 0.9$, $B_2 = 0.98$, $\epsilon =$ 1e-6 and weight decay set to 0.01. Batch size per worker is set to 64, so that with 32 total workers and 2x gradient accumulation it resulted in effective batch size of 4096.

One session of training takes about 80 hours on 32 Nvidia A100 (80Gb) GPUS (4 8-GPU nodes)

which amounts to 2560 GPU-hours.

As downstream tasks I use a number of datasets from GLUE (MNLI Williams et al. (2018) with adversarial HANS McCoy et al. (2019b) validation set, SST2 Socher et al. (2013b), QQP, MRPC Dolan and Brockett (2005b), COLA) and SUPERGLUE suites (RTE Bentivogli et al. (2009), BoolQ Clark et al. (2019a)) as well as stand-alone datasets like SQuAD_v2 Rajpurkar et al. (2016c).

For sequence classification tasks I use a linear layer as classification head while for question answering tasks I use a custom implementation (adding additional output unit to predict if the question is answerable, and a custom implementation for answer scoring logic)

I also investigate models with siamese architecture, which have been successful on NLI (Williams et al., 2018) and in conjunction with BERT encoders (Reimers and Gurevych, 2019).

Figure 4.8 illustrates performance between pre-training and fine-tuning epochs. In some cases even well pre-trained models marginally benefit from additional fine-tuning, but at earlier stages this effect is more pronounced.

Another interesting aspect that I notice is that different tasks behave differently. Some tasks such as QQP (Figure 4.8b) achieve close to SOTA performance at as little as 5% of pre-training process, while "harder" tasks like RTE keep improving to as much as I did pre-train, and seemingly could keep improving further.

Figure 4.9 displays best validation accuracy for each pre-training stage for a wider range of tasks.

CHAPTER 5

Language Models at Work: Technology Extraction from Patents

This Chapter focuses on applications of Language Models to the extraction of technological knowledge from patents. Part of the content of this Chapter has been used in the writing of Puccetti et al. (2021a, 2023).

The analysis and observation of technologies is a fundamental part of technological management, especially for planning R&D policies by governments and companies (Yoon and Park, 2005). Given the present pace of innovation, market actors need a deeper understanding of a technological domain, to mitigate the uncertainty typical of the digital ages (Robinson et al., 2013).

In the literature there exist many methods for technological mapping and forecasting, that can help decision makers in predicting core and emerging technologies (Huang et al., 2021), diffusion of technologies (Daim et al., 2006), convergence (Karvonen and Kässi, 2013) and portfolio analysis (Ernst, 2003). Patents are among the best source of information to reach these goals (Joung and Kim, 2017) because they contain rich technical details. Anyway, they are also among the most complex textual sources to analyse automatically because of the mix of technical and juridical jargon.

A patent provides the technical description of an invention in a sufficiently clear and complete manner to enable a person skilled in the art (i.e. someone having the relevant technical information publicly disclosed at the time of the invention) to replicate the invention without any additional creative activity (Art. 83 EPC) (Lidén and Setréus, 2011). A patent is designed to disclose the minimum content that makes understandable and reproducible the invention. The use of juridical jargon makes the document legally binding, eligible to protect the invention.

Patent texts create a barrier to the access to one of the widest technical open access resources. It is believed that between 70% and 90% of the information about technologies can only be found in patents (Asche, 2017). Patent analysis is a valuable approach for deriving information about an industry or technology for forecasting (Daim et al., 2006), competitive analysis (Thorleuchter et al., 2010), technology trend analysis (Tseng et al., 2011), and for avoiding infringement (Yu and Zhang, 2019). This information may be obtained by the use of either bibliometric analysis and text mining. The former involves the analysis of meta-data, such as citations, assignee, inventors, and International Patent Classification classes (Cho and Kim, 2014). The latter aims to extract relevant information from unstructured text, that includes title, abstract, claims, state-of-the-art description, and other records (Tseng et al., 2007).

Given the complexity of patent texts, bibliometric analysis is more common in literature since it involves the analysis of well organized and structured data, more easily processable than unstructured data. However, the analysis of meta-data is not able to capture the detailed technical content of patents (Kuhn et al., 2020; Righi and Simcoe, 2019). As affirmed by Vicente-Gomila et al. (2021), text mining enhances traditional measures based on bibliometric data in forecasting technological change.

Recently, it has been shown that text mining is more effective than metadata analysis for measuring novelty and impact of a patent. In Arts et al. (2021), the authors use text mining techniques to identify new technologies and measure patent novelty, detecting uni-grams (single word, well-known as keywords), bi-grams and tri-grams (two or three consecutive words, also called keyphrases) in title, abstract, or claims of a patent. They consider as emerging technologies a word or a words' combination appearing for the first time in the text, achieving remarkable results with generic keywords and keyphrases. Other studies involve text mining from patent for identifying emerging technologies (Ranaei et al., 2020; Zhou et al., 2020; Porter et al., 2019; Jang et al., 2021; Sarica et al., 2020) or exploring the convergence phenomena among technological fields (Gustafsson et al., 2015; Song et al., 2017). At the state-of-the-art, text mining techniques for technological analysis focus on generic terms and not on specific ones for investigating the technological change. This is a limitation given the quantity and quality of technical information contained in patents, therefore novel approaches are needed to enable these contents.

In this Chapter, I use Natural Language Processing (NLP) to recognize the technologies mentioned in a patent, thus overcoming this gap in literature. I reach this goal by tackling a well known task in NLP called Named Entity Recognition (NER) using Transformer-Based Language Models. NER systems are widely used to extract general entities (such as persons' names, cities, dates and times), but there is still a lack of tools able to extract technically relevant information (Fantoni et al., 2013; Chiarello et al., 2018a, 2020, 2021). I measure the performances of three different NER methods in terms of precision, recall and computational time.

The three methods I test are lexicon-based, rule-based and distributional NER. Lexicon-based and rule-based NER exploit predefined lists of entities (lexicons) or rules-driven experts systems. Distributional NER exploits transformer-based language models to learn which technologies to extract. In the present Chapter I use the state-of-the-art for entity extraction, Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019).

The tool for identifying technologies from text proposed in this Chapter has two main contributions. It contributes to the field of scientometrics and technology analysis, overcoming the gap in literature related to the analysis of patents for understanding the technological change. It gives a contribution to practitioners, especially policy makers and companies, by supporting them in patent analysis to recognize emerging technologies, map technological convergence or investigate the content of a patent.

The results of this work show how the methods proposed extract $4,731$ technologies in a set of $1,600$ patents with a precision of about 40%. I also analyse in detail all methods proposed and show that some are more fit to search for small numbers of specific entities (lexicon), whereas others are more effective in gathering large numbers of technologies with less control over which they retrieve (rules, distributional methods).

The most novel result of this work is that it provides a first attempt to recognize technologies from texts, avoiding the collection of technological-related terms that are not technologies, such as function, user, advantage or drawback entities. Moreover, this work extracts each type of technologies, overcoming the gap in the existing literature, that is focused only on the collection of emerging technologies.

In the following Section, I present all the data and the methodologies used. Afterwards, I show the results in Section 5.2.

5.1 Data Collection and NER Methodologies

I employ NLP systems to identify technologies in the title, abstract, claim and state-of-the-art description of the patents. I develop three different approaches based on lexicons, rules and distributional methods. In this section, I describe each of them and outline their advantages and drawbacks. Throughout the rest of the chapter, in order to ease the reading, I will occasionally refer to the items (technologies in this case) extracted as *entities*.

I set the experiment employing four case studies to measure the performance of the different methods proposed in this paper. The database chosen for retrieving the patent documents is the Erre Quadro S.r.l.[1] one. The database contains over 90 million patents and includes high quality bibliographical and legal status patent information from leading industrialized and developing countries. The Erre Quadro database was chosen due the fact that it indexes patents from different European and International databases, e.g., European Patent Office (EPO) and World Intellectual Property Organization (WIPO). The proposed method may also be replicated using other patent database, such as the United States Patent and Trademark Office (USPTO).

In literature, the common strategies for retrieving patents are based on keywords (Maghrebi et al., 2011), patent classification systems (e.g. International Patent Classification system, IPC) (Ozcan and Islam, 2017) or their combination (Park et al., 2013). I define three retrieving principles in order to simplify the measurement task:

- **Perform a IPC-based searching strategy**: The IPC is a hierarchical classification of patents based on the technological area. Each classification code is composed of 7 characters. The first character is a letter, it is called *section*, and indicates the technical field. Then, there are two numbers, called *class*, that indicate the subject matter of the patent. The next letter constitutes the *subclass*, i.e. the units in which the *classes* are divided. Afterward, there is the File Index classification (FI), a three-digit number (*IPC subdivision symbol*) and/or one letter (*file discrimination symbol*), that represents a *main group* or *subgroup* and indicates the theme of the patent (OECD, 2009). I choose to perform an IPC search because this classification well determines the boundary of each class, as established in the Strasbourg Agreement of 1971. So, since I aim to develop a system for automatically recognizing technologies in a given technological field, having well-defined technological domains eases measuring how much each extraction system is precise (precision) and complete (recall).

- **Select the first 5 digits for each IPC**: The number of digits selected for each IPC class is 5 for a balanced identification of technological content, avoiding to consider too generic or too fine grained technologies as explained in Section 2.8;

- **Collect different IPC section**: I choose to consider different IPC sections (namely first letter of the IPC code) for performing the most extensive and exhaustive comparisons among the various methods. In fact, the section is the broadest patents classification level.

I selected the following IPC codes:

- **A63F 3/00** Board games, Raffle games;

- **C23C 2/00** Hot-dipping or immersion processes for applying the coating material in the molten state without affecting the shape;

- **E04B 5/00** Floors; Floor construction with regard to insulation;

- **H04J 1/00** Frequency-division multiplex systems.

Whereas the chosen *subclass* codes are the following:

- **A63F** CARD, BOARD OR ROULETTE GAMES;

- **C23C** COATING METALLIC MATERIAL;

- **E04B** GENERAL BUILDING CONSTRUCTIONS;

- **H04J** MULTIPLEX COMMUNICATION.

[1] Available: https://www.errequadrosrl.com/, Accessed: July 28, 2021.

I search for the higher diversity in terms of languages and domains and select cases where the inventions is user-centered (A63F, E04B) or not (C23C, H04J); where technical subject matter may represent a product (A63F), an apparatus (H04J) or a process (C23C, E04B); where technical subject matter may be a physical device (A63F, C23C, E04B) or an electronic/software system (H04J). The selection process is fundamental for identifying difference among the methods developed in this study and understanding which biases are embedded in each one.

For each selected IPC code, I retrieve also the patents set of the relative *subclass* (e.g. the *subclass* of A63F 3/00 class is A63F). These data will be fundamental for training the distributional NER system as I explain later in this Section. The *subclass* is more general than the complete IPC code and includes diversified patents, so it can help in the identification of different technologies. This leads to improve the models with reference to the recall, but it may produce minor losses regarding the precision. I will show this is not a dramatic drawback, indeed for one of the methods it leads to a small improvement for both measures.

In the rest of the paper, I refer to the complete IPC code as *IPC group* and the related *IPC subclass* as *IPC category* to avoid misinterpretations.

For each IPC category and IPC group I select 400 patents. Regarding the IPC category, I choose this number to have a data-set that is a good fit for the training of the models. On average I find patents to contain around 300 sentences. A classical benchmark data-set for NER (Tjong Kim Sang and De Meulder, 2003) contains around 14,000 sentences for training, therefore I select this amount of patents in order to reach a comparable amount. I gather a larger number of sentences closer to 100,000 from this patent set. Indeed, I need to compensate the lower precision of rules as opposed to human tagging: a larger amount of sentences helps average out part of the imperfections in the data.

I choose 400 patents for the IPC group as well for the sake of comparison. Although I analyse the number of entities extracted as well, my focus is on a fair comparison among the methods. I need to make sure that I do not give an advantage to the trained methods by providing them broader sets of technologies, thus I need IPC categories and IPC groups to be equal in number. Moreover, this does not pose any limitation on the patent set validity.

5.1.1 Lexicon-based NER

The lexicon-based method is the simplest available, and it consists in using lexicons of technologies. For the selection of the sources I rely on the work by Giordano et al. (2023) that uses freely available online sources:

- ***Wikipedia***: the Wikipedia contributors list a set of emerging and potentially disruptive technologies[2]. It contains 397 different technologies. In Bonaccorsi et al. (2020), it is demonstrated that Wikipedia's list of emerging technologies has a degree of coverage in the range 90%–95% with respect to academic journals, consultancy reports and specialized blogs;

- ***O*NET***[3]: it is an occupational framework developed by the U.S. Department of Labor which is made of 974 occupations classified on the basis of the Standard Occupational Classification (SOC) system and their corresponding skills, knowledge, abilities and technologies. The framework includes 30,173 different technologies. O*NET divides those technologies in (i) 21,267 machines and equipment (70.48 %) and (ii) 8,906 information, communication and software technologies (29.52 %).

The use of lexicons allows to work in a controlled environment, where one can know in advance the detectable type of entities. Therefore, given the lexicons source and specifications, I focus only on a particular kind of technologies.

This solution is extremely fast to deploy and can be used to inspect several thousands of documents in very short time. Looking for the flaws of this approach, the biggest one is the difficulty in gathering high quality lexicons containing large numbers of technologies. The cleansing of a lexicon is performed by highly trained people, so such a process is hard to scale to several thousands of entities.

[2] Available: https://en.wikipedia.org/wiki/List_of_emerging_technologies, Accessed: July 28, 2021.
[3] Available: https://www.onetonline.org/, Accessed: July 28, 2021.

Furthermore, the lexicons used in this work aim to identify only a part of technologies mentioned in patents. The Wikipedia list contains only emerging technologies. These last are well defined in literature and have five main characteristics: (i) radical novelty, (ii) relatively fast growth, (iii) coherence, (iv) prominent impact, and (v) uncertainty and ambiguity (Rotolo et al., 2015). The O*NET lexicon derives from an occupational framework. So, it includes the technologies related to job occupations with a coarse-grained level of detail, mentioning only the commercially available technologies. Moreover, there is an unbalanced focus on information and software technologies (more than 20 % of the total), reflecting the current situation of job landscape, where the digital transformation had large effects (Frey and Osborne, 2017).

Beyond this, lexicons are hard to adapt to different contexts, as demonstrated also in biomedical domain by Odat et al. (2015). Patents are an extremely broad data-set with all kinds of technologies and a lexicon able to cover this variegated type of knowledge would need to contain a vast amount of entities, hard to develop and to maintain as well. Furthermore, the lexicon NER suffers from ambiguity (Carlson et al., 2009), where words like *python* can have different meanings (e.g. an animal or a programming language) depending on the context.

Given the above considerations, pre-built lexicons aim to minimize (i) the human effort in listing all possibly technologies, and (ii) the machine time to recognize technologies in text. However, this methodology will likely extract a smaller number of entities with some blurred technologies.

5.1.2 Rule-Based NER

The second family of NER systems I employ is based on rules. In a general sense, this means I define morphosyntactic patterns that identify the presence of a technology in a patent text. This type of extraction can be made in several ways and there is literature about it (Hearst, 1992; Roller et al., 2018; Chiarello et al., 2018b). This methodology, if properly adapted to each case, turns out to be quite effective especially in terms of coverage.

To outline the methodology with an example, I will use the hypernym/hyponym relation indicating more and less abstract concept. An example of such relation is the one between phone and smartphone. Indeed, whenever something is identified as a smartphone it can be identified also as a phone and at the same time there are phones that are not smartphones.

I use two different rule-based methods:

- ***Extractor 4.0***: regular expressions for extracting technologies 4.0 developed in Chiarello et al. (2018b);
- ***Hearst*** regular expressions for searching hypernym/hyponym relation (Roller et al., 2018).

The first rule-based approach consists of a list of automatically generated regular expressions aimed at extracting Industry 4.0 terms. Regular expressions are pattern matching queries that can be built to match virtually any pattern. More complex patterns require for more complex queries and this poses a limit to the generality they can achieve. Nevertheless, the list built in (Chiarello et al., 2018b) contains about 1,600 expressions and covers several different cases, making it a valuable extraction tool.

I expect an high application field dependence due to the fact that this approach was originally developed for Industry 4.0. On the other hand, the flexibility of regular expressions and the data driven way used to develop them give it good generality, therefore I expect it to show higher recall than lexicons.

The rules based on hypernym/hyponym relation instead are derived from Hearst (1992), it builds upon the methodology that assigns to each word its role in the sentence (Petrov et al., 2012), e.g. nouns, indicated as *NOUN* or adjectives as *ADJ*.

I search for *NOUN* tokens and then I use fixed patterns as described in Hearst (1992) to select the surrounding specifiers. These patterns are developed to match hypernym/hyponym pairs. However, they do not automatically identify technologies. I add a step to find the technologies, indeed, I keep only pairs including one of the following words as their hypernym: *technology, machine, device, apparatus, mechanism, sensor, network, system, unit*.

The selected words identify general terms that are hypernyms to large families of technologies. To choose them, I use the work from Jang et al. (2021) where the authors define words such as *system, unit, device, apparatus* and so on as high level concepts in terms of automotive technology.

The choice of words introduces a bias into this methodology making it lean towards more machinery and tools based fields. This will affect its ability to perform in certain IPC groups.

I expect a decrease in precision in the use of rules compared to the lexicon, since I use not too strict patterns to extract a relevant amount of entities. On the other hand, the recall of this method, namely the fraction of technologies extracted, is supposed to increase.

5.1.3 Distributional NER

I train several Transformer-based NER models, using a version of BERT (Devlin et al., 2019) to obtain contextual representations of text, a 2 layer bidirectional Long Short Term Memory (LSTM) and conditional random fields. I call these methods distributional as in this context they are mostly interesting since, unlike the two other methodologies (lexicon and rules) which are humanly curated, these algorithms are trained.

Similar to the techniques used in Chapter 4 to asses the performance of language models, the approach used here is composed of pre-training and fine-tuning. In particular, the model I use to encode text is BERT which has been trained on a large amount of textual data. I perform additional training on the task at hand, namely technology identification, adding a small number of trainable parameters. Therefore all the analysis I show about the performance and the results obtained by this approach add to the previous analysis of language models, although in the specific context of patents.

This approach leads to near state of the art performance on NER benchmarking datasets such as CoNLL-03 data-set (Tjong Kim Sang and De Meulder, 2003). Moreover, these models need to be trained on tagged data. This is done using the set of patents extracted from the categories for each IPC.

For each IPC group, namely A63F 3/00, C23C 2/00, E04B 5/00 and H04J 1/00, and for both rules extractor, namely *Extractor 4.0* and *Hearst*, I build a different extractor based on distributional methods. Each of these methods will be called ***Dist Hearst*** or ***Dist 4.0*** respectively. Notice how I omit the IPC group name to ease the notation and in the rest of the paper which IPC group is under study is explicitly stated.

Let us describe the pipeline used to build each extractor. The process is as follows:

1. select a IPC group and a NER method;
2. take the associated IPC category, for example A63F for A63F 3/00, as described at the beginning of this Section;
3. annotate the patents in this IPC category using the method I selected;
4. save the output which is now a set of sentences with tagged technologies;
5. train the distributional model on this set of sentences;
6. Finally, use this trained model to extract from the IPC group selected in the first step.

Beyond this approach, for both *Hearst* and *Extractor 4.0*, I train one model following the same steps outlined above with the difference that each of the category associated with an IPC group is substituted with a patent set built randomly selecting 100 patents from each of the IPC category. The models obtained will be called ***Dist Hearst All*** and ***Dist 4.0 All***. Summing up:

- ***Dist 4.0*** is the distributional method using the *Extractor 4.0* NER trained on IPC cateogry;
- ***Dist Hearst*** is the distributional method using the *Hearst* NER trained on IPC cateogry;
- ***Dist 4.0 All*** is the distributional method using the *Extractor 4.0* NER trained on a set of 100 randomly selected patents from each of the IPC category;
- ***Dist Hearst All*** is the distributional method using the *Hearst* NER trained on a set of 100 randomly selected patents from each of the IPC category.

My hypothesis is that the wider range of technologies available in the higher level IPC, together with the adaptability of rule based methodologies, will lead to a high quality training set. In turn the generalization skills of contextual word embedding should be able to make use of a data-set encompassing such knowledge in order to learn from it in a proficient way. I believe that, once trained, these models can be used in the IPC groups to extract an higher number of technologies compared to rules.

I expect an improvement in recall, whereas it is hard to make predictions about precision, since the more general IPC group could be both an improvement or a drawback from this point of view.

I choose to not manually check the training set for constructing the distributional methods for two reasons. First of all, the manual check is time consuming and expensive: it requires a domain expert to read each technology extracted by the rule-based NER carefully. Second, I aim to observe the achievable levels of precision and recall of distributional methods in condition of low effort in terms of time and cost.

In parallel, I do not perform any information based cleaning (e.g. tf-idf filtering) for two reasons. First, custom pre-processing would influence the results. Second, this process would require an assessment on how it is performed (per IPC, per corpora, per method, etc.). In addition, just like manual filtering, it constitutes a large effort on its own. However, I believe that this might be useful for improving a model performance in future works.

The implementation of the distributional methods was done using the Python package *Flair*, developed in (Akbik et al., 2018). The training is performed on a Quadro RTX 6000 with 24 gigabytes of memory. The same parameters are kept over all experiments, the batch size is fixed at 64, for the recurrent neural network I use a hidden size of 256 which was found to be good for all cases, as embeddings I use BERT-base and a learning rate of $5 * 10^{-3}$ for the training that lasted 20 epochs. The high computational cost of the experiments limited the possibility to make an exhaustive hyper-parameters search. I tried some preliminary experiments changing learning rate that didn't lead to significant variations in performance and I thus chose a single set of parameters for all models. In the rest of this work I identify which approach among the several tested is best and in future works, focusing on a single model I will optimize it further through a large set of possible parameters.

5.1.4 Performance Evaluation

The list of technologies recognized by the NER systems is manually checked to compute precision and recall. In literature, the results of several NLP tasks have traditionally been evaluated using human subjects (Belz and Reiter, 2006) or using previously annotated data (Lee et al., 2020b). As discussed in Section 2.8, in the technological domain there is a lack of annotated documents. For these reasons, I rely on human evaluation in a manner similar to Giordano et al. (2023). The validation was performed in a straightforward way by two PhD students in Smart Industry at the University of Pisa. both students are provided with a table containing the list of extracted entities for each IPC group (the same for both students). The assignment was "*Read each extracted entity and decide whether the entity is a technology or not in the context where it appears, comparing each entity with the Definition 1 of technology, provided in Subsection 2.9.1*".

I remark how this task is itself complex, since all the entities, even those that are not technologies in a strict sense, belong to close semantic fields. Several ambiguous cases are found, for example *high frequency signal*, which to a non expert might appear as a technology, is a physical behaviour. According to Definition 1, one can see that this example is too generic. On the contrary the similar entity *high frequency signal interface* is an example of technology and the noun *"interface"* helps disambiguating it with respect to the physical behaviour.

This case demonstrates how the human validation is itself very challenging. Indeed, from an algorithmic perspective, the two aforementioned entities are extremely similar, since they differ in only one word, while being mostly composed of rare words, making it hard to infer that the fourth one is what makes the difference (Bernier-Colborne and Langlais, 2020).

For understanding the degree of agreement among the independent observers who assess the completed list of the technologies the inter-rate agreement is calculated using the Fleiss' Kappa (Fleiss, 1971). A set of 300 technologies is provided to 5 people and the same instructions described above for evaluating whether the entity extracted is a technology or not. I stress the fact that the 5 authors have a different background (1 Mechanical Engineer, 2 Management Engineers, 1 Mathematician and 1 Aerospace

Engineer) and experience (1 Professor, 1 Researcher, 3 PhD students), adding value to the evaluation of the technologies because it is carried out by experts from different perspectives. The inter-rate agreement is 0.516 and the authors give the same rating for the 50.30% of the technologies in the sample. Moreover, 4 authors agree on an additional 29.00% of the samples and the remaining 20.70% meet the agreement of only 3 out of 5 authors, highlighting how complex is evaluating when a concept is a technology.

Several studies in literature offer rules of thumb for interpreting the inter-rate agreement. Many authors agree that an inter-rate agreement value greater than 0.50 is the target condition to be confident on the results (Landis and Koch, 1977; Fleiss et al., 1981; Regier et al., 2013). It is possible to reach higher values with more complex evaluations. For example, Zhang et al. (2020b) attempt to build a fine-grained entity annotation corpus of clinical items, manually annotating more than $10,000$ clinical records in five rounds. In each round, 100 records are randomly selected, and the inter-rate agreement is calculated: the initial value of 0.40 increased step by step reaching 0.94 at the fifth round (Zhang et al., 2020b). In this case, one review round is performed, reaching 0.516 inter-rate agreement. This result, higher than the initial value presented in Zhang et al. (2020b), shows a moderate but confident agreement.

An iterative process of annotation can help in improving the inter-rate agreement, especially for engineering a system able to recognize technologies in textual data. However, the goal here is to demonstrate that is possible to use NER methods for extracting technologies from textual data and to identify which NER system perform better than others.

5.2 Technological NER Results

In Subsection 5.2.1 to Subsection 5.2.3, I provide insights about the performances of the different approaches used to extract technologies from patents, I analyse qualities and flaws of each method and I study how they work together. In Subsection 5.3.2, using the four IPC groups outlined in Subsection 5.1.4, I describe how these methods allow researchers and practitioners to map a given knowledge domain. Table 5.1 summarises the different methodologies to which I refer from now on as reported in the column *Method*.

Type	Method	Explanation
Gazetteer	Wikipedia	List of emerging technologies from Wikipedia.
	O*NET	List of technologies from the occupational framework O*NET.
Rules	Extractor 4.0	Regular expressions for extracting technologies 4.0.
	Hearst	Regular expressions for searching hypernym/hyponym relation.
Distributional	Dist 4.0	Distributional method using the Extractor 4.0 NER trained on each IPC cateogry using BERT.
	Dist Hearst	Distributional method using the Hearst NER trained on each IPC cateogry using BERT.
	Dist 4.0 All	Distributional method using the Extractor 4.0 NER trained on all the IPC cateogries using BERT.
	Dist Hearst All	Distributional method using the Hearst NER trained on each IPC cateogries using BERT.

Table 5.1: *Summary of methods per types with explaination*

The total number of unique entities extracted by the NER systems is $12,572$ in a set of $1,600$ patents. After the human validation process described in Subsection 5.1.4, the portion of entities that are considered technologies by the revision is 38 % ($4,731$ different technologies).

This score is in line with the similar work of Giordano et al. (2023), which attempts to identify technologies from 300,000 patents on defence sector, reaching an overall precision of 35.39 % and collecting $1,090$ different technologies. However, the score is low if compared with other NER systems; but, as mentioned in Section 2.9, the absence in literature of a training set for technologies negatively affects the performance of the extraction systems.

Let me report insights about the number of technologies per patent. The median of extractions per patent is 81 technologies (counted with repetition) and 10 distinct ones. So, on average per sentence there are 0.35 technologies and 0.04 distinct ones. This means that, one needs to read about 3 sentences to encounter a technology and about 25 sentences for a new unseen one in a patent.

These numbers, although only averages, highlight that patents are generally densely populated with technologies while only containing few unique ones. The difficulty of technologies extraction emerges: this tasks requires to cherry-pick the few interesting ones within each patent.

Based on the work of Chiarello et al. (2018a), I use the following metrics for the evaluation of the whole technologies extraction process:

- *Training time*: time needed to create the statistical model using the training set in the distributional-based NER;
- *Extraction time*: time needed to extract the entities on the patent set;
- *Precision*: number of technologies that a system correctly detected in textual data divided by the total number of entities identified by the system;
- *Relative recall*: proportion that the NER system retrieves of the total technologies retrieved by all systems together for a given IPC group.

In particular, I measure the precision and recall based on the human validation. As described in Subsection 5.1.4, evaluating the recall is a challenging task for a domain where an annotated corpus in not available, as in this case. For this reason, I relied on relative recall, instead of traditional recall. Sampson et al. (2006) define relative recall as the proportion that any NER system retrieves of the total technologies retrieved by all systems considered to be working as a composite.

For the scope of this analysis, these metrics allow me to consider both the effectiveness (Precision and Relative recall) and efficiency (Training time and Extraction time). The latters are evaluated for a good reproducibility of this work. The training time is suitable for researchers and practitioners who want to develop their own distributional NER methods for recognizing technologies in textual data. The extraction time mainly helps practitioners that want to apply this NER methods for their own purposes. In general, the measure of training and extraction time allows to better estimate the duration of the analysis and plan the related tasks. Databases, such as Hadoop and Spark, can be easily integrated with the proposed methodology, resulting in a time reduction in the training of distributional methods or in the extraction of technologies from the text. These frameworks are widely adopted in current academic and business landscape, however they require expertise and knowledge to set up the architectures to process the data.

In Table 5.2 I compare all the metrics across all methodologies and IPC groups. I report also the number of unique entities automatically extracted in a patent set (Found) and the number of distinct entities considered as technologies after the manual review (Correct). These values are used in the computation of Precision and Relative Recall. In the next paragraph, I describe and discuss each class of methods presented in Section 5.1.

5.2.1 Performance of Lexicon-Based NER

The lexicons achieve the highest precision over all tasks, as shown in Table 5.2. They are hand crafted and thus this result is not unexpected. However, they provide a smaller number of entities when compared to the other methodologies, as the recall points out.

The *O*Net* lexicon is the most consistent method in terms of found technologies (Correct) across different IPC groups. As pointed out in Section 5.1, this result is due to its structure, where almost all descriptions (70.48%) contain names of machines and equipment. It behaves more coherently than rule-based approaches or distributional methods, though at the cost of a lower amount of retrieved technologies (Correct).

The *Wikipedia* lexicon has lower performances in terms of Relative Recall than the other methods. The amount of found technologies (Correct) is lower than 15 per IPC group. However, one may consider that *Wikipedia* lexicon includes only 397 different technologies, while *O*Net* has more than 30,000 technologies. Although the sizes of two lexicon differ by more than 2 orders of magnitude, the number of Found and Correct technologies differs only by an order of magnitude. This may be due to the fact that the technologies in *O*NET* are indicated with commercial names, rare to find in patents.

5.2.2 Performance of Rule-Based NER

The effectiveness of rule-based NER is domain dependent. Both *Hearst* and *Extractor 4.0* suffer a performance loss on the IPC groups C23C 2/00 and E04B 5/00 in terms of both precision and recall. In

IPC	Training Time	Extraction Time	Method	Found	Correct	Precision	Relative Recall	F1
A63F 3/00 Board Games	-	3s	O*Net	286	148	0.517	0.152	0.235
		2s	Wikipedia	27	13	0.481	0.013	0.024
		732s	Extractor 4.0	547	284	0.519	0.292	0.373
		348s	Hearst	398	216	**0.543**	0.222	0.315
	900m	1,720s	Dist 4.0	771	378	0.490	0.388	0.433
		1,714s	Dist Hearst	2,362	743	0.315	**0.764**	0.446
	500m	1,729s	Dist 4.0 All	732	375	0.512	0.385	0.439
		1,721s	Dist Hearst All	1,063	414	0.389	0.425	0.406
C23C 2/00 Coating solutions	-	3s	O*Net	302	178	**0.589**	0.318	0.413
		2s	Wikipedia	31	10	0.323	0.018	0.034
		745s	Extractor 4.0	865	123	0.142	0.220	0.172
		353s	Hearst	274	96	0.350	0.171	0.229
	600m	1,702s	Dist 4.0	301	118	0.392	0.211	0.274
		1,685s	Dist Hearst	791	183	0.231	**0.327**	0.270
	500m	1,706s	Dist 4.0 All	305	127	0.416	0.227	0.294
		1,702s	Dist Hearst All	605	171	0.283	0.305	0.293
E04B 5/00 Floors construction	-	3s	O*Net	283	184	**0.650**	**0.399**	0.494
		2s	Wikipedia	17	5	0.294	0.015	0.028
		771s	Extractor 4.0	494	71	0.144	0.211	0.171
		376s	Hearst	313	62	0.198	0.185	0.191
	500m	1,858s	Dist 4.0	267	76	0.285	0.226	0.252
		1,837s	Dist Hearst	548	86	0.157	0.256	0.194
	500m	1,811s	Dist 4.0 All	254	81	0.319	0.241	0.275
		1,849s	Dist Hearst All	442	79	0.179	0.235	0.235
H04J 1/00 Multiplex systems	-	5s	O*Net	248	174	**0.702**	0.071	0.128
		3s	Wikipedia	22	13	0.591	0.005	0.010
		1,010s	Extractor 4.0	1,332	528	0.396	0.215	0.278
		527s	Hearst	1,019	640	0.628	0.261	0.368
	800m	2,674s	Dist 4.0	1,634	930	0.569	0.379	0.455
		2,671s	Dist Hearst	3,132	1,651	0.527	**0.673**	0.591
	500m	2,583s	Dist 4.0 All	1,597	906	0.567	0.369	0.447
		2,563s	Dist Hearst All	1,373	680	0.495	0.277	0.355

Table 5.2: *Performance of proposed NER systems*

general, all rule-based methods perform better on H04J 1/00 than on any other IPC group. These results may be explained by the design of those methods:

(i) *Extractor 4.0* developed by Chiarello et al. (2018b) aims at identifying Industry 4.0 technologies, by definition referred to "cyber-physical-system" (Lu, 2017), that are more related with electronic and software technologies, as in the case of H04J 1/00 (Frequency-division multiplex systems) and also for A63F 3/00 (Board games, Raffle games);

(ii) *Hearst* method derived from Hearst (1992) was designed to recognize modular technologies, namely technologies structured in parts, sub-parts, systems, devices and so on.

Therefore, IPC groups like C23C 2/00 and E04B 5/00, rooted in processes and constructions, are farther apart from the focus of these tools. Whereas H04J 1/00 and A63F 3/00 better fit the features of *Hearst* and *Extractor 4.0*, as confirmed by the performance over these IPC groups. These results may reflect the biases embedded in the adopted methods that uphold the choice to select different case studies (IPCs), as explained in Section 5.1.

5.2.3 Performance of Distributional NER

Different behaviours demonstrated by the distributional methods root on different reasons I explain hereinafter.

The performance of the rules-based methods reflects on the performance of the distributional models trained on them. That is, if the rules perform worse on a certain IPC group the distributional methods will behave the same way. However, I notice an improvement of the precision in IPC group C23C 2/00. This is likely due to higher precision and recall of the extractions in the IPC category. Indeed, it is hard to understand which patterns lead to a decision about a technology selection since distributional methods are inherently difficult to explain (Pedreschi et al., 2019).

Let me also underline the most unexpected behaviours in Table 5.2. The high number of technologies found by *Dist Hearst* in H04J 1/00 is noteworthy. In general this group has higher number than all others in terms of retrieved technologies (Correct), the highest number of extracted entities (Found) among IPC groups, and the highest precision (Precision) among the others as well. One more characteristic of this IPC is that, even though *Extractor 4.0* and *Dist 4.0* detect a comparable amount of entities ($1,332$ and $1,634$, respectively), *Extractor 4.0* achieves this with a lower precision. Such a behaviour is not easy to interpret given the black box nature of distributional models (Pedreschi et al., 2019).

One more general trend is that rules, particularly *Hearst*, are only partially outperformed by distributional methods. The recall is higher in some IPC groups (H04J 1/00 and A63F 3/00). However, the time consumption is about three times longer than Hearst (without considering the training time).

The recall of distributional methods is what makes it worth training them. Despite not providing improvements in terms of precision, they increase the recall in all IPC groups. This means that they are able to recover a larger amount of technologies in the same amount of text. This difference depends on the strictness of rules, which only identify specific patterns, opposed to the flexibility of distributional methods, that, though highly data dependent, can learn to identify any pattern.

One more information conveyed by Table 5.2 is that the methodology developed by training on a merged set of patents, *Dist Hearst All* and *Dist 4.0 All*, are also a valid approach. Indeed, they achieve similar precision and recall, they are often within 0.05 variation in precision and 0.1 variation in recall over all IPC groups (except for H04J 1/00) compared to the respective *Dist* method despite being trained only once.

Regarding the time costs, *Dist Hearst All* and *Dist 4.0 All* provide a faster method since they need to be trained only once with a training time equal to 500 minutes, instead of one time in each IPC groups, where the overall time of the training is 2,800 minutes (900 for A63F 3/00, 600 for C23C 2/00, 500 for E04B 5/00 and 800 for H04J 1/00). Furthermore, the performance of these methods is good on all IPC groups, though partially worse than those trained in the respective categories, *Dist Hearst* and *Dist 4.0*.

5.2.4 Global vs Local Training of Distributional NER

Looking at Table 5.2 as a whole one can see that the differences in precision and recall of distributional methods among IPC groups support the choice to select a diversified data-set, as I further investigate in this section.

One limitation of this work is that I trained distributional models on a tagged data-set created using rules. For example, *Dist Hearst* was trained on a data-set tagged by *Hearst*, thus one can expect a large overlap between the technologies found by *Hearst* on the IPC category A63F and by *Dist Hearst* on the IPC group A63F 3/00.

I provide a measure of the fact that though this dependence exists, there is a limited overlap between the technologies found in the categories and those found in the IPC groups. In Table 5.3, I show for both *Dist Hearst* and *Dist 4.0* how many (in absolute and percentage values) of the technologies found in the groups by distributional methods (Dist Hearst, Dist 4.0) are also found by the rules in the IPC categories (Category Hearst, Category Extractor 4.0).

As one can see for all IPCs and for all methods, a large portion of the extractions in the IPC group was not previously found in the IPC category, indeed, in all cases less than 50% and in several cases below 30% of the entities are found by the rules. This supports the validity of this methodology, in particular, it makes the training effort worthy. In this table one can also find better performances in IPC group H04J

IPC	Dist Hearst	Category Hearst (%)
A63F 3/00	743	307 (41)
H04J 1/00	**1,651**	330 (20)
C23C 2/00	183	48 (26)
E04B 5/00	86	26 (30)
Total	*2,264*	*589 (26)*
	Dist 4.0	Category Extractor 4.0 (%)
A63F 3/00	378	177 (47)
H04J 1/00	**930**	233 (25)
C23C 2/00	118	43 (36)
E04B 5/00	76	23 (30)
Total	*1,198*	*298 (25)*

Table 5.3: *Performance of Global vs Local Training*

1/00, the reasons are the same as outlined above and they depend on biases of the algorithms related to the patent IPC groups.

5.3 Comparing NER Methodologies

Table 5.4 shows the number of shared technologies per pair of NER systems, the diagonal of the table reports the number of technologies found by each extractor.

I expect that the maximum number of technologies contained in the patents, though this amount is unknown, influences the level of intersection among different methodologies. Therefore, as a general remark, Table 5.4 can indicate the complementarity among the methodologies.

The distributional methodologies show a good level of intersection among them because of the high number of extracted entities and the training processes executed on similar data. Each of them has a stronger intersection with the rules used to build their respective training data. For example, the number of shared entities is 377 for the pair *Dist Hearst* and *Hearst* and 203 for *Dist Hearst* and *Extractor 4.0*. Therefore, the rules used to tag the training data of the distributional approaches leads to higher intersection between the respective couple of methods. In addition, the distributional methods appear consistent because they are trained on the same data-set, albeit tagged differently. Therefore, they register high intersection values.

O*Net	454							
Wikipedia	3	22						
Extractor 4.0	38	9	660					
Hearst	73	6	86	922				
Dist 4.0	51	10	311	206	1,211			
Dist Hearst	130	11	203	377	565	2,303		
Dist 4.0 All	50	9	253	192	658	532	1,158	
Dist Hearst All	68	4	100	209	318	573	339	1,051
	O*Net	Wikipedia	Extractor 4.0	Hearst	Dist 4.0	Dist Hearst	Dist 4.0 All	Dist Hearst All

Table 5.4: *Pairwise comparison of proposed NER methods*

On the contrary, methods based on lexicons show little overlap with other methodologies. As a meaning of example, the *O*Net* lexicon retrieves a significant amount of entities, though lower than the rules, but only shares a little part of them with the other methods. *O*Net* and *Dist Hearst* register an apparently high overlap, i.e., 130 over 454 technologies found by *O*Net*. However, this number is not as meaningful as it may appear at first given the large quantity of technologies found by *Dist Hearst*.

One of the leads that can be inferred from Table 5.4 is that a combination of different methodologies results in a higher number of retrieved technologies. Indeed, a large amount of technologies are only found by specific methodologies, though there is a non negligible part of shared entities. This supports the initial claim: a blended extractor, resourcing to a suite of approaches, is a viable solution and appears to be the best one.

Table 5.5 reports the number of entities found by each subset of methods. The technologies identified by more than one approach are counted only in the cases that contain all those methods. For example, *microsoft windows* is found by rules and lexicons, so it is counted in the sixth case and not in the second nor in the third.

The highest intersection is between rules and distributional methods. It may depend on the fact that these two methods provide the most numerous among all extractions.

A large share of the extractions performed by the lexicons are only found by them (285), underlining again the ability of this solution to provide few high quality results, as expected by a humanly curated list.

The least intersection is among *Rules* and *Lexicons* (17), while the amount of technologies shared by all methods is higher (91). Therefore, I can infer that when a technology is found by both *Lexicon* and *Rules* than it is also found by *Distributional* methods. A consequence of this analysis is that one could consider giving up rules for the sake of simplicity, whereas lexicons appear to provide good improvements compared to how much information they recover.

In Table 5.5 I also report a qualitative comparison among the technologies found by different families of methodologies. I provide a number of examples of the extractions to underline how different methodologies are highly complementary.

Cases	Lexicon	Rules	Distributional	Amount	Examples
1			✓	2,870	aircraft, air motor, antenna feeder, card game machine, card shuffler, digital assistant, digital cable, optical transport network, computer hard drive, radio access node, ram memory, reactor pump, relay system, remote control bar, rf circuit
2		✓		749	adsl transceiver, air vent, air vent hole, antenna port, integrate circuit lead frame, information communication technology, pentium ii, pc network station, proximity detector, record playback apparatus, repeater, rfid tag, robot gun, satellite broadcast center, server machine
3	✓			285	air lift pump, alarm system, conveyor system, allen wrench, ball valve, bridge crane, external hard drive, fire alarm system, flowmeter, ground table, pressure transducer, refrigeration unit, roulette wheels, smoke detector, tape recorder
4		✓	✓	639	airplane, antenna, central processing unit, access terminal, adsl, barcode, bluetooth, cellular network device, coaxial cable, lte network, magnetic tape, smart tv, touchscreen, transistor, tv camera
5	✓		✓	80	boiler, centrifuge, computer workstation, control valve, disk, earphone, freezer, mass storage device, motorcycle, optical filter, plasma screen, truck, vacuum chamber, web server, wireless communication system
6	✓	✓		17	air conditioner, air knife, decoiler, deflector, hemodynamic monitor, inclinometer, microsoft windows, pacemaker, php, pointing device, radiant heater, signal generator, track ball, water jet cutter, wearable computer
7	✓	✓	✓	91	defibrillator, desktop computer, card reader, forklift, javascript, linux, mainframe computer, microprocessor, optical sensor, pager, robotic arm, router, slotmachine, touch screen, workstation

Table 5.5: *Quantitative Comparison of the different NER systems* ***Note:*** *The table shows how many entities are found by each subset of the technologies, the first three columns indicate which families are considered in each row and the fourth the number of entities found by them, finally the fifth shows some examples belonging to the respective group of extractors.*

Certain specific technologies (e.g., *air lift pump*) are only found by lexicons, whereas other simpler ones either only by rules (e.g., *air vent*), or only by distributional methods (e.g., *air motor*). There is also an opposite pattern, that is a general entity found by the rules and by distributional methods (e.g., *antenna*) is then generalized through the latter, identifying new entities (e.g., *antenna feeder*), though not all of them (e.g., *antenna port*).

This example highlights again how each methodology is helpful to the final goal of technologies identification and that a blended solution using all the strategies attempted would be the best.

One can notice also that the technologies detected by all methods (last row of Table 5.5) belong in large part to electronics, as expected by the strongest performance all solutions have on IPC group H04J 1/00.

In addition, the extracted entities present all types of morphological patterns. Among the examples reported in Table 5.5 I find:

- *pentium ii, linux, javascript:* proper nouns;
- *truck:* common noun;
- *vacuum chamber, air conditioner:* noun + noun;
- *fire alarm system, card game machine:* noun + noun + noun;
- *digital assistant, smart tv:* adjective + noun;
- *remote control bar:* adjective + noun + noun;
- *computer hard drive:* noun + adjective + noun;
- *integrate circuit lead frame:* adjective + noun + adjective + noun.

I remark that beyond the few examples reported in Table 5.5, inspecting all the extractions, I have seen that for entities shorter than three words all patterns are found in large numbers and high quality, concerning entities that are 4 words long or more, they are rarer but keep a consistent quality.

5.3.1 Efficiency of the NER Systems

I conclude the quantitative analysis of the results discussing the efficiency of the NER systems. Figure 5.1 reports the number of correctly found technologies in each IPC group by each method versus the time it took to perform the extraction.

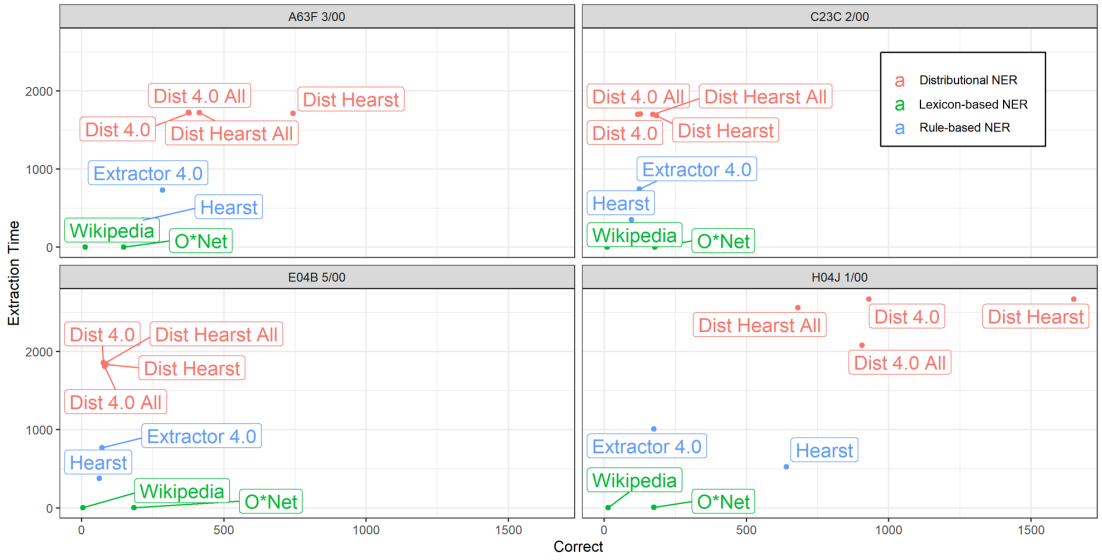


Figure 5.1: *Number of Correct Technologies vs Extraction Time for IPC group*

In general, all the methodologies are coherent with reference to extraction time. The lexicons are the fastest approach, *Extractor 4.0* and *Hearst* are placed as second best, and all the Dist methodologies appear to be the slowest. These results are consistent with the number of technologies each solution finds, though not directly dependent.

A question that comes to mind looking at these results is whether the value provided by distributional methods is worth their extra time cost. Different approaches can be used for different tasks. In the case of emerging technologies mapping, distributional methods can lead to better performances, since on average they are able to find more technologies. For mapping more stable sector, lexicon approaches can be more

efficient, since they require less manual revision. In cases of high level studies involving large corpora of patents (Arts et al., 2021; Jang et al., 2021; Giordano et al., 2023), the distributional approach can make the time costs prohibitive for tagging the technologies in the text, while adding a limited amount of significant information. Indeed, a broader limitation of *Distributional* methods is the higher effort in development phase than *Lexicon* or *Rules*.

Beyond time consumption, Figure 5.1 shows another interesting fact. The performance of the distributional methodologies depends on the performance of the method they are based on, as observable by the horizontal components of the figure. I discuss earlier the reasons for this dependence, here the goal is to highlight the magnitude. For instance, *Dist Hearst* depends on *Hearst*. Indeed, whenever *Hearst* retrieves a larger number of technologies (Correct) in a class with respect to another, a larger improvement in the number of technologies found by *Dist Hearst* is registered. To see this, one can notice how between IPC groups A63F 3/00 and H04J 1/00 there is an increase of 423 in the number of Correct for *Hearst*, while for *Dist Hearst* the increase is 908, and a similar behaviour happens whenever *Hearst* increases. This is relevant since it hints that larger experiments could lead to a strong gain in terms of collected technologies. In addition, as previously mentioned, the better performance of *Hearst* in A63F 3/00 and even more in H04J 1/00 are due mainly to the set of words used as general technologies' hypernyms, which are a good match for these two IPC groups related to tools and machinery. On the contrary, for IPC groups C23C 2/00 and E04B 5/00, dealing with processes, the chosen words perform more poorly.

5.3.2 Benchmarking Trough Comparative Assessment

In this section, I attempt to compare the proposed methodologies to previous studies in literature. As discussed in Section 2.8, Jang et al. (2021) develop a method to extract technological information from textual data of patents, aiming at investigating a given technological domain. Keeping in mind the different goals between this work and my analysis, I replicate their methodology. To show a qualitative comparison on the extractions obtained with the two approaches (i.e., the methodologies presented here and the one in Jang et al. (2021)). Note that I only replicate the part of the work from Jang et al. (2021) that can be compared to what is done in this Chapter, namely the vocabulary construction, and that in this work there isn't a human evaluation on this set of technologies.

I analyse the most recurring technologies and the number of patents where they are found for each IPC group to compare the type of the extracted technologies and their frequency. Figure 5.2 reports the results of my approach and Figure 5.3 depicts the ones obtained through the method developed in Jang et al. (2021).

These approaches are able to identify field specific technologies. For example, in Figure 5.2 one can find *valve* in C23C 2/00 (in the top right chart), defined as "Hot-dipping or immersion processes for applying the coating material in the molten state without affecting the shape" which involves working with liquids; or *clip* in E04B 5/00 (in the bottom left chart), described as "Floors; Floor construction with regard to insulation" and thus involving constructions. Anyway, some entities are found in other less suited IPC groups. For instance, *level*, which is a good match for E04B 5/00 (in the bottom left chart), is found also in 71 patents of A63F 3/00 (in the top left chart). However, since the revision performed manually takes into account field specificity in this IPC group it is removed. These results can prove the validity of choosing diversified IPC groups and the generality of the methods proposed, which are able to detect field specific items though with different performances on different IPC group. Some general entities are found as well in many IPC groups and they may be not informative. As a meaning of example, *computer* is listed in three charts in Figure 5.2 since it is found in all IPC groups except C23C 2/00. However, those general terms can not be removed a priori because they do represent technologies although in a very general sense. The loss of term specificity in some cases and the extraction of too general technologies are both limitations of this work which I intend to further investigate.

The IPC group H04J 1/00 registers the highest value of occurrences, the second is A63F 3/00 and last are C23C 2/00 and E04B 5/00. The large margin of H04J 1/00 can be explained with the biases in the employed methodologies, as already discussed.

The terms reported in the charts of Figure 5.3, such as *steel* in C23C 2/00 (in top right chart) and *base station* in H04J 1/00 (in bottom right chart), appear very general. Comparing Figure 5.3 and Figure 5.2, one can highlight the specificity of the technologies detected in most IPC groups by the present solution

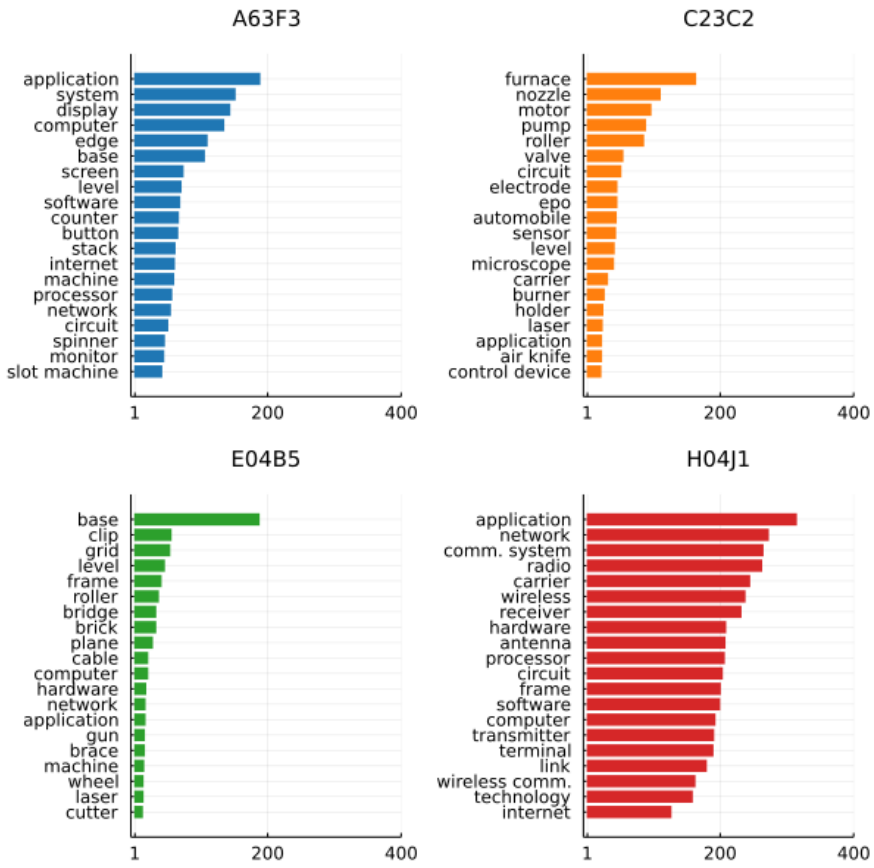


Figure 5.2: *Top 20 frequently technologies for each IPC group using the approach proposed here.*

(e.g. *brick* in E04B 5/00 and *pump* in H04J 1/0). This behaviour is due to the higher specificity imposed on the methods developed here, using the words shown in Section 5.1, *system*, *unit*, *device*, *apparatus* and so on.

A direct implication of this aspect is the difference in the number of occurrences of words, since more specific terms appear more rarely than more general ones. Indeed, the number of occurrences is always lower for my extractions except for IPC groups H04J 1/00, where the top 20 extractions are more general for the methods I propose as well and thus the occurrences are closer. The solutions therefore differ in terms of generality and specificity of lexical information extracted. Let me remark how this difference is not negative for neither of the works, indeed it pushes each one towards its goal and does not make an alternative one of the other.

In Figure 5.3, I find as well terms that don't represent technologies. Words like *player* in A63F 3/00, *support* in E04B 5/00, *comprise* in C23C 2/00 and *number* in H04J 1/00 are not technologies on their own. Indeed, the goal of Jang et al. (2021) is to develop a lexicon of technology related terms. So that the identification of *player* in the patents of the IPC group A63F 3/00 concerning board and raffle games is a good result from their point of view. Anyway, this results does not fit my goal of identifying technologies. These considerations motivates the large difference in the type of terms.

The compared analysis underlines the choices made by the two works and so the respective purposes. An example of how these two methodologies can be used in combination for different aspects of the same problem is as follows: the studies on the evolution of technological domain require a wide perspective enabling a high level analysis of the changes it undergoes over time. In these cases, Jang et al. (2021) is able to create a broad map of the concepts used in the description of the relevant entities.

The analysis on targeted technologies over selected sets of patents demands for a fine grained approach. Thus, the methodologies proposed in this Chapter are more suited for an in-depth study on a certain

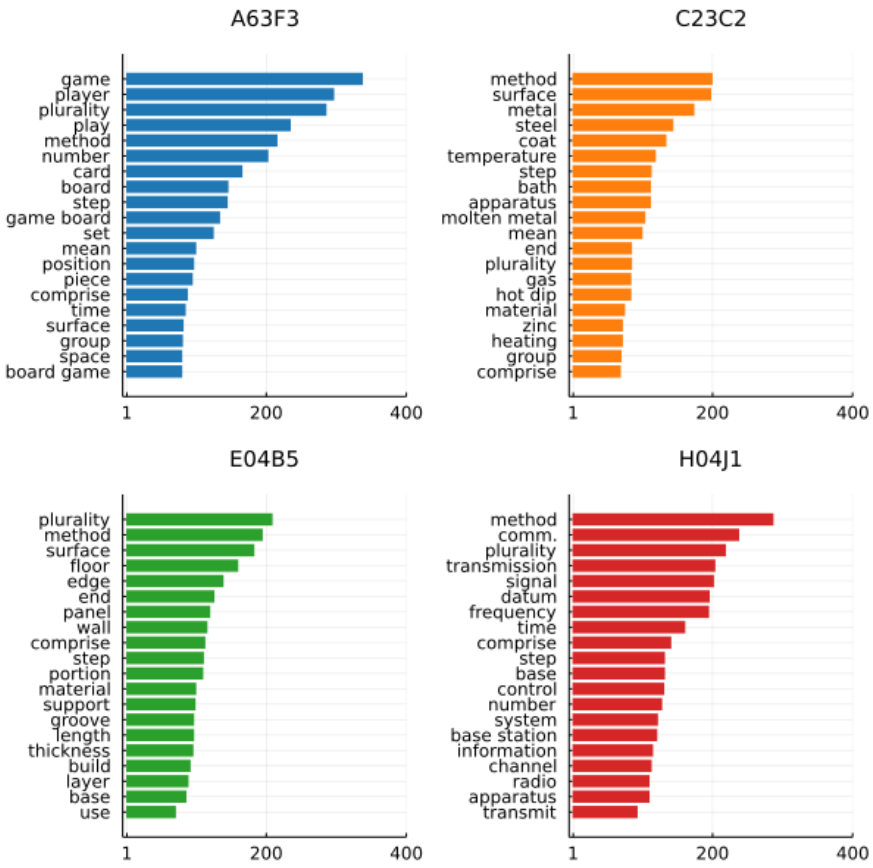


Figure 5.3: *Top 20 frequently found entities for each IPC group using the approach of Jang et al. (2021)*

technology. Therefore, I believe these two solutions can work in parallel towards the challenging goal of technology-related information extraction from patents.

CHAPTER 6

Conclusions

This thesis proposes to address the lack of understanding of Transformer Based Language Models, mostly due to the vast amount of parameters that compose them, by analyzing models such as BERT and RoBERTa with the goal to deepen the understanding of their inner mechanisms. It proposes to do so from several perspectives, measuring the linguistic competence of such models, as well as analyzing special (outlier) parameters together with training dynamics. Finally, since this model family has become widely used in practical scenarios, I expand the analysis ground from general-purpose texts and linguistic skills to more technical corpora such as patents, where I make a parallel analysis of the application efficacy and model properties in such diverse domain.

In Chapter 3, I propose an in-depth analysis aimed at understanding how BERT embeddings encode and organize linguistic competence. Relying on a variable selection approach applied to a suite of 68 probing tasks. I show the existence of a relationship between the implicit linguistic knowledge encoded by the LM and the number of individual units involved in the encoding of this knowledge. I find that, according to the strategy for obtaining sentence-level representations, the amount of hidden units devised to encode linguistic properties varies differently across BERT layers: while the number of non-zero units used in the *Mean-pooling* strategy remains approximately constant across layers, the *[CLS]* representations show a continuous increase in the number of used coefficients. Moreover, I notice that this behavior is particularly significant for linguistic properties related to the whole structure of the syntactic tree. At the same time, features belonging to POS and dependency tags tend to require fewer non-zero units across layers (Puccetti et al., 2021b). Finally, I find that it is possible to identify groups of units more relevant to specific linguistic tasks. In particular, I show that by clustering the set of sentence-level properties according to the weights assigned by the regression models to each BERT unit one can identify clusters of features referable to the same linguistic phenomena, despite some variations, this is valid for both the configurations and for all the BERT layers. This analysis of linguistic properties leads to discover the existence of special entries relevant to most linguistic tasks, similar to phenomena that have also been found by others in parallel (Kovaleva et al., 2021; Luo et al., 2021), generally referred to as outliers.

In Chapter 4, I study the outliers phenomenon in Transformer-based models (in particular BERT and RoBERTa) and try to identify the causes and effects of this phenomenon. I find that the magnitude of hidden state dimensions corresponding to outliers correlates with the vertical self-attention pattern, which enables attention to focus on the classification tokens. Furthermore, I find that there are two types of outliers: some of them affect Masked Language Modeling performance the most in the middle layers

(where the correlation with token frequency is also at its peak), and for others, the impact grows towards the final layers (even though the correlation with token frequency decreases). These findings suggest that outliers are due not to the Transformer architecture per se, but rather to the highly skewed token frequency distribution in textual pre-training data (Puccetti et al., 2022). In that case to mitigate outliers, one might need to design a pre-training scheme that better accounts for such distributions. Outliers have been further investigated in other works (Bondarenko et al., 2021; Dettmers et al., 2022; Dettmers and Zettlemoyer, 2023), to understand how they develop during pre-training while affecting the performance of fine-tuned models. This is relevant since parameters' quantization, used to reduce the memory needed to run larger models, suffers from outliers, which can not be treated similarly to other parameters and need special care.

After focusing on BERT-like models' performance and mechanics on a number of tasks meant to measure their language understanding ability, I apply variants of this model that are pre-trained and fine-tuned on patents, to understand how they behave in more challenging settings.

Chapter 5 offers a viable Named Entity Recognition system based on rules, lexicons and machine learning techniques (Puccetti et al., 2023). The system attempts to extract technologies from patent documents, providing quantified metrics. These metrics are based on traditional measures, namely precision and recall. I estimate the time to perform each extraction to support researchers and practitioners in understanding the advantages and limitations of this system. I test the method using four case studies collected with a patent classification system-based (IPC) search strategy. For each field of analysis, I show the ability of the NER system in unveiling the most cited technologies. The proposed method may be a valid technology identification tool that can be integrated or used in conjunction with other text analysis pipelines to support academics and industrial actors in investigating a technological domain. The system developed is able to collect $4,731$ technologies from $1,600$ patents. The model outperforms previous literature in terms of precision and recall (Giordano et al., 2023). Moreover, I compare the results of the technology extraction process with the work of Jang et al. (2021). This allows to analyze a technological domain at a fine-grained level, avoiding the noise brought by generic terms. As pointed out in Section 5.2, the NER system enables the identification of an average of 0.35 technologies per sentence (approximately 1 technology every 3 sentences) and a median of 10 distinct technologies per patent. Therefore, technologies are not rare words in patents if compared with other recognizable entities, such as users (Puccetti et al., 2021a), advantages/drawbacks (Chiarello et al., 2017), affordances (Chiarello et al., 2019) and biases (Melluso et al., 2021).

To understand the working of transformers in this challenging domain, I have compared the performance of NER methods built upon these models with others based on lexicons and rules with a twofold goal. On one side, by measuring scores and costs at the same time to understand the value of these models in applications, on the other by manually inspecting their output to identify possible systematic errors that would go unnoticed by automatic approaches.

Moreover, I compare the effectiveness of all these methods in different patent classes and measure their generalization and out-of-distribution abilities, adding to the previous analyses of their inner workings. I show that these models have improved generalization skills across different patent classes together with their ability to recover a larger amount of technologies compared with more classical methods, despite those being based on domain expertise provided by knowledgeable creators.

From an industrial point of view, the system developed in this thesis may improve the analysis of technological domains to map the landscape and forecast the diffusion of technologies. Traditional methods for the identification of competitors and partners using paper documents may benefit from this system to include additional information in the analysis. For example, a technology extraction process may be involved in the text mining pipeline of Vicente-Gomila et al. (2017) to consider the competitors also from a technological point of view. Fareri et al. (2020) use text mining to identify industry 4.0 technologies and estimate the impact of the fourth industrial revolution on the workforce of a multinational company. Similarly, practitioners may employ the method developed here for the analysis of technological competencies to assess the need for re-skilling or up-skilling at the company level.

Despite the fast pace at which the field of Natural Language Processing moves on, the analyses I carried out are relevant as they shed light on the inner mechanisms that make these models work. Indeed, while models have grown in number of parameters, the architecture and training strategies have remained almost unchanged, making my conclusions applicable to newer ones. Indeed, outliers, which I investigated in BERT and RoBERTa, have been found in models of all scales and qualities, including the most recent

state of the art ones.

Therefore, the results of this thesis pose the ground for a better understanding of the transformer architecture workings. The same holds for application in patents, which, after the promising results in this work, can benefit from larger models and newer techniques but still need to take into account the adaptability to patent classes and more in general need careful assessments from human evaluators for a fruitful deployment.

6.1 Limitations

Let me describe the most notable limitations I see in this work. The first one is that during the development of this thesis, as mentioned in Section 4.2, models have grown in size at a remarkable speed, therefore the results obtained for smaller ones need a reassessment in newer larger ones. This is a complex task as very large models are more difficult to handle from an engineering perspective.

Secondly, probing tasks, while useful in assessing the linguistic knowledge of large language models, do not fully reflect such models' language understanding, which is instead better measured by humans, this is a possible further development that in this work has only been done for the patent domain but would also be relevant and informative in a language understanding setting.

Concerning the study of outliers in language models, the strongest limitation is the lack of a theoretical reason for the presence of such large out-of-distribution parameters. While I do find evidence that relates this phenomenon to the tokens distribution, others prove this dependency in special cases (Gao et al., 2019) and there are several studies concerning similar properties (Wei et al., 2021), there isn't a formal proof for why this happens.

For the study on the patent domain, the strongest limitation lies in the precision the models achieve. Although I manage to extract a large number of technologies, only about 40% of the extracted entities are identified as correct by the manual evaluation, while in other fields (Fan et al., 2020) higher performances are achieved. The low value can be understood considering two main aspects. First, I did not use any manually tagged gold training set because, to the best of my knowledge, there are no examples of such a set for technologies in patents.

The solution used for developing a training set poses difficulties to improve beyond a certain threshold. The development of a gold set is a challenge for future work that I intend to tackle, learning from the findings of this one. In parallel, I also mention the absence of a proper evaluation of the recall. Though the large number of extracted technologies and having double checked them makes me confident that I extract a non-trivial share of the knowledge available in the analyzed patents, its time cost prevents from a manual evaluation of the share of technologies extracted from a patent.

I also remark that relying on search and retrieval of patent records from the patent database may influence the results because of the access conditions of the database. Indeed, there are not many examples of freely available databases. Excepting very few (notably, Freepatentsonline[1]), most patent databases require licensing or give limited access. And this is probably one of the sources of boundaries in this line of research.

6.2 Future Steps

To make an assessment of which future steps are most promising one needs to confront the recent developments of NLP. The most relevant one, that touches most aspects of this field, is the development of generative Large Language Models (LLM) trained on causal language modeling. As described in Subsection 2.3.1 these models are pre-trained on generating the upcoming tokens one after the other. Remarkably, scaling such models to hundreds of billions of Parameters and training them on trillions of tokens, as shown in Subsection 2.4.2, opens new possibilities for few-shot learning Radford et al. (2019) and zero-shot learning Brown et al. (2020) and this developments influence most aspects of my thesis.

However, while the capabilities of newer, larger, generative models are remarkable, these models are built similarly to the ones I study. In Chapter 2, I describe several variants of language modelings underlining how, while they are used on different applications and with different results, they all share a

[1] Available: `https://www.freepatentsonline.com/`, Accessed: July 28, 2021

large amount of both technical and conceptual building parts. Indeed, analyses performed on one training approach (Masked Language Modeling) carry to the other (Causal Language Modeling). Many results of my thesis can be developed further in light of these recent developments, without loosing relevance, let me describe those I believe are the most prominent new directions.

Firstly, staying the closest to the work so far, scaling indentical experiments to larger generative models encompassing large amounts of parameters (hundreds of billions). Indeed, these models have shown several emergent properties (Wei et al., 2022a) that appear only beyond a certain model scales. These are novel phenomena that need studying through the investigation of models of varying scale up to and beyond 30 billion parameters, providing new challenges that come with training and testing such large models, note that this is already being done by other works (Dettmers et al., 2022) with findings matching those I show.

A different way to go beyond the present study is investigating language models able to process images together with text (Yu et al., 2022; Li et al., 2023; Wang et al., 2022). Adding a new modality makes model inspection more challenging and requires the development of new benchmarks and techniques, e.g. linguistic probing needs to be paired with tasks able to assess the role of images in the development of linguistic knowledge. Most interestingly, for future analyses, the majority of these models gain an advantage from creating a common representation space where similar images and texts are mapped closely. This poses a promising step towards the grounding of textual information into other media, images in this case. Indeed, the availability of such similar representation spaces allows the application of methodologies such as linguistic probing "to images" and in turn, this could lead to the identification of properties of visual data that can be used proficiently to generate text (as humans do).

A third direction is the involvement of humans in the model inspection and assessment process, since in this thesis this approach is only explored for the patent domain, where the kind of review I perform involves knowledge of the field, rather than the assessment of the output quality from a human/linguistic perspective. The development of large generative models, makes this perspective relevant, as the performance of general purpose generative models is harder to assess compared to task-specific ones similar to those I investigate.

Concerning applications to patents, there are also several developments to tackle in future works, that involve humans. In this case, annotators could help in the development of high-quality data-sets to train models. One possible method for saving time and cost in the creation of a training set using rule-based NER is a human evaluation of the training set in the form of Active Learning. The expert checks the quality of a sample of the initial training data-set obtained with the rule-based NER (i.e., 30% of the training data-set can be used) and remove terms that are not technologies. Then, the distributional NER is trained on the sample of the training data-set for a quality check. The quality of the distributional method is then measured, if it reaches an acceptable quality one stops the training, otherwise, the same methodology can be re-applied to more data until a certain degree of training data quality is reached. Afterward, the development of a test set will enable a more precise assessment of the recall of the systems. In the patent context as well, generative models open the path towards zero-shot NER as they can be queried in natural language to extract knowledge from patents' passages exposing a previously impossible way to carry out Technological NER in patents.

Bibliography

Abbas, A., Zhang, L., and Khan, S. U. (2014). A literature review on the state-of-the-art in patent analysis. *World Patent Information*, 37:3–13.

Abujabal, A., Saha Roy, R., Yahya, M., and Weikum, G. (2018). Never-ending learning for open-domain question answering over knowledge bases. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1053–1062, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Adi, Y., Kermany, E., Belinkov, Y., Lavi, O., and Goldberg, Y. (2017). Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *International Conference on Learning Representations*.

Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Arts, S., Hou, J., and Gomez, J. C. (2021). Natural language processing to identify the creation and impact of new technologies in patent text: Code, data, and new measures. *Research Policy*, 50(2):104144.

Asche, G. (2017). "80% of technical information found only in patents" – is there proof of this [1]? *World Patent Information*, 48:16–28.

Ba, L. J., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *CoRR*, abs/1607.06450.

Baan, J., Leible, J., Nikolaus, M., Rau, D., Ulmer, D., Baumgärtner, T., Hupkes, D., and Bruni, E. (2019). On the realization of compositionality in neural networks. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 127–137, Florence, Italy. Association for Computational Linguistics.

Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12449–12460. Curran Associates, Inc.

Bau, A., Belinkov, Y., Sajjad, H., Durrani, N., Dalvi, F., and Glass, J. (2019). Identifying and controlling important neurons in neural machine translation. In *International Conference on Learning Representations*.

Belinkov, Y. and Glass, J. (2019). Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Belinkov, Y., Màrquez, L., Sajjad, H., Durrani, N., Dalvi, F., and Glass, J. (2017). Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Beltagy, I., Lo, K., and Cohan, A. (2019). SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Belz, A. and Reiter, E. (2006). Comparing automatic and human evaluation of nlg systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Bentivogli, L., Magnini, B., Dagan, I., Dang, H. T., and Giampiccolo, D. (2009). The fifth PASCAL recognizing textual entailment challenge. In *Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*. NIST.

Bernier-Colborne, G. and Langlais, P. (2020). HardEval: Focusing on challenging tokens to assess robustness of NER. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1704–1711, Marseille, France. European Language Resources Association.

Blanco-Fernández, Y., Gil-Solla, A., Pazos-Arias, J. J., Ramos-Cabrer, M., Daif, A., and López-Nores, M. (2020). Distracting users as per their knowledge: Combining linked open data and word embeddings to enhance history learning. *Expert Systems with Applications*, 143:113051.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Blevins, T., Levy, O., and Zettlemoyer, L. (2018). Deep rnns encode soft hierarchical syntax. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 14–19.

Bonaccorsi, A., Chiarello, F., Fantoni, G., and Kammering, H. (2020). Emerging technologies and industrial leadership. a wikipedia-based strategic analysis of industry 4.0. *Expert Systems with Applications*, 160:113645.

Bondarenko, Y., Nagel, M., and Blankevoort, T. (2021). Understanding and overcoming the challenges of efficient transformer quantization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7947–7969, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Breitzman, A. and Thomas, P. (2015). The emerging clusters model: A tool for identifying emerging technologies across multiple patent systems. *Research policy*, 44(1):195–205.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Brunato, D., Cimino, A., Dell'Orletta, F., Venturi, G., and Montemagni, S. (2020). Profiling-ud: a tool for linguistic profiling of texts. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 7147–7153, Marseille, France. European Language Resources Association.

Caragea, D., Chen, M., Cojoianu, T., Dobri, M., Glandt, K., and Mihaila, G. (2020). Identifying fintech innovations using bert. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1117–1126. IEEE.

Carley, S. F., Newman, N. C., Porter, A. L., and Garner, J. G. (2018). An indicator of technical emergence. *Scientometrics*, 115(1):35–49.

Carlson, A., Gaffney, S., and Vasile, F. (2009). Learning a named entity tagger from gazetteers with the partial perceptron. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*, pages 7–13.

Carroll, L. S. L. (2017). A comprehensive definition of technology from an ethological perspective. *Social Sciences*, 6(4).

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. (2017). SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Strope, B., and Kurzweil, R. (2018). Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Chang, H.-S. and McCallum, A. (2022). Softmax bottleneck makes language models unable to represent multi-mode word distributions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8048–8073, Dublin, Ireland. Association for Computational Linguistics.

Chang, S.-H. and Fan, C.-Y. (2016). Identification of the technology life cycle of telematics: A patent-based analytical perspective. *Technological Forecasting and Social Change*, 105:1–10.

Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Wang, Z., and Carbin, M. (2020). The lottery ticket hypothesis for pre-trained bert networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15834–15846. Curran Associates, Inc.

Chiarello, F., Belingheri, P., Bonaccorsi, A., Fantoni, G., and Martini, A. (2021). Value creation in emerging technologies through text mining: the case of blockchain. *Technology Analysis & Strategic Management*, pages 1–17.

Chiarello, F., Bonaccorsi, A., and Fantoni, G. (2020). Technical sentiment analysis. measuring advantages and drawbacks of new products using social media. *Computers in Industry*, 123:103299.

Chiarello, F., Cimino, A., Fantoni, G., and Dell'Orletta, F. (2018a). Automatic users extraction from patents. *World Patent Information*, 54:28–38.

Chiarello, F., Cirri, I., Melluso, N., Fantoni, G., Bonaccorsi, A., and Pavanello, T. (2019). Approaches to automatically extract affordances from patents. In *Proceedings of the Design Society: International Conference on Engineering Design*, volume 1, pages 2487–2496. Cambridge University Press.

Chiarello, F., Fantoni, G., Bonaccorsi, A., et al. (2017). Product description in terms of advantages and drawbacks: Exploiting patent information in novel ways. In *DS 87-6 Proceedings of the 21st International Conference on Engineering Design (ICED 17) Vol 6: Design Information and Knowledge, Vancouver, Canada, 21-25.08. 2017*, pages 101–110.

Chiarello, F., Trivelli, L., Bonaccorsi, A., and Fantoni, G. (2018b). Extracting and mapping industry 4.0 technologies using wikipedia. *Computers in Industry*, 100:244 – 257.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Cho, Y. and Kim, M. (2014). Entropy and gravity concepts as new methodological indexes to investigate technological convergence: Patent network-based approach. *PloS one*, 9(6):e98009.

Choi, S., Lee, H., Park, E., and Choi, S. (2022). Deep learning for patent landscaping using transformer and graph embedding. *Technological Forecasting and Social Change*, 175:121413.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. (2021). Rethinking attention with performers. In *International Conference on Learning Representations*.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. (2022). Palm: Scaling language modeling with pathways.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. (2019a). BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. (2019b). What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Clark, K., Luong, M., Le, Q. V., and Manning, C. D. (2020). ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Conneau, A., Kruszewski, G., Lample, G., Barrault, L., and Baroni, M. (2018). What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Daim, T. U., Rueda, G., Martin, H., and Gerdsri, P. (2006). Forecasting emerging technologies: Use of bibliometrics and patent analysis. *Technological forecasting and social change*, 73(8):981–1012.

Dalvi, F., Durrani, N., Sajjad, H., Belinkov, Y., Bau, A., and Glass, J. (2019). What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6309–6317.

Dalvi, F., Sajjad, H., Durrani, N., and Belinkov, Y. (2020). Analyzing Redundancy in Pretrained Transformer Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4908–4926, Online. Association for Computational Linguistics.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Re, C. (2022). Flashattention: Fast and memory-efficient exact attention with IO-awareness. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

de Marneffe, M.-C., Simons, M., and Tonhauser, J. (2019). The commitmentbank: Investigating projection in naturally occurring discourse. *Proceedings of Sinn und Bedeutung*, 23(2):107–124.

De Rassenfosse, G., Dernis, H., Guellec, D., Picci, L., and de la Potterie, B. v. P. (2013). The worldwide count of priority patents: A new indicator of inventive activity. *Research Policy*, 42(3):720–737.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. (2022). GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

Dettmers, T. and Zettlemoyer, L. (2023). The case for 4-bit precision: k-bit inference scaling laws.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dolan, B. and Brockett, C. (2005a). Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*. Asia Federation of Natural Language Processing.

Dolan, W. B. and Brockett, C. (2005b). Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Dong, Y., Cordonnier, J.-B., and Loukas, A. (2021). Attention is not all you need: pure attention loses rank doubly exponentially with depth. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2793–2803. PMLR.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Ernst, H. (2003). Patent information for strategic technology management. *World patent information*, 25(3):233–242.

Ethayarajh, K. (2019). How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

Fan, C., Wu, F., and Mostafavi, A. (2020). A hybrid machine learning pipeline for automated mapping of events and locations from social media in disasters. *IEEE Access*, 8:10478–10490.

Fantoni, G., Apreda, R., Dell'Orletta, F., and Monge, M. (2013). Automatic extraction of function–behaviour–state information from patents. *Advanced Engineering Informatics*, 27(3):317–334.

Fareri, S., Fantoni, G., Chiarello, F., Coli, E., and Binda, A. (2020). Estimating industry 4.0 impact on job profiles and skills using text mining. *Computers in industry*, 118:103222.

Fernández, N., Arias Fisteus, J., Sánchez, L., and López, G. (2012). Identityrank: Named entity disambiguation in the news domain. *Expert Systems with Applications*, 39(10):9207–9221.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Fleiss, J. L., Levin, B., Paik, M. C., et al. (1981). The measurement of interrater agreement. *Statistical methods for rates and proportions*, 2(212-236):22–23.

Foundation, T. W. (2022-06-01). Wikipedia, the free encyclopedia. `https://en.wikipedia.org`. Accessed: 2022-06-01.

Frey, C. B. and Osborne, M. A. (2017). The future of employment: How susceptible are jobs to computerisation? *Technological forecasting and social change*, 114:254–280.

Gao, J., He, D., Tan, X., Qin, T., Wang, L., and Liu, T. (2019). Representation degeneration problem in training natural language generation models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. (2021). The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027.

Gildea, D. (2001). Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*.

Giordano, V., Chiarello, F., Melluso, N., Fantoni, G., and Bonaccorsi, A. (2023). Text and dynamic network analysis for measuring technological convergence: A case study on defense patent data. *IEEE Transactions on Engineering Management*, 70(4):1490–1503.

Goldberg, Y. and Hirst, G. (2017). *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers.

Gordon, A., Kozareva, Z., and Roemmele, M. (2012). SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398, Montréal, Canada. Association for Computational Linguistics.

Gordon, M., Duh, K., and Andrews, N. (2020). Compressing BERT: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 143–155, Online. Association for Computational Linguistics.

Gustafsson, R., Kuusi, O., and Meyer, M. (2015). Examining open-endedness of expectations in emerging technological fields: The case of cellulosic ethanol. *Technological Forecasting and Social Change*, 91:179–193.

Hain, D. S., Jurowetzki, R., Buchmann, T., and Wolf, P. (2022). A text-embedding-based approach to measuring patent-to-patent technological similarity. *Technological Forecasting and Social Change*, 177:121559.

Han, E. J. and Sohn, S. Y. (2015). Patent valuation based on text mining and survival analysis. *The Journal of Technology Transfer*, 40(5):821–839.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

He, P., Liu, X., Gao, J., and Chen, W. (2021). Deberta: Deconding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*.

Hewitt, J. and Liang, P. (2019). Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Hewitt, J. and Manning, C. D. (2019a). A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Hewitt, J. and Manning, C. D. (2019b). A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., Driessche, G. v. d., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. (2022). Training compute-optimal large language models.

Hofmann, P., Keller, R., and Urbach, N. (2019). Inter-technology relationship networks: Arranging technologies through text mining. *Technological Forecasting and Social Change*, 143:202–213.

Hohman, F., Park, H., Robinson, C., and Polo Chau, D. H. (2020). Summit: Scaling Deep Learning Interpretability by Visualizing Activation and Attribution Summarizations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1096–1106.

Hossari, M., Dev, S., and Kelleher, J. D. (2019). Test: A terminology extraction system for technology related terms. In *Proceedings of the 2019 11th International Conference on Computer and Automation Engineering*, pages 78–81.

Huang, Y., Zhu, F., Porter, A. L., Zhang, Y., Zhu, D., and Guo, Y. (2021). Exploring technology evolution pathways to facilitate technology management: From a technology life cycle perspective. *IEEE Transactions on Engineering Management*, 68(5):1347–1359.

Jain, S. and Wallace, B. C. (2019). Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Jang, H., Jeong, Y., and Yoon, B. (2021). Techword: Development of a technology lexical database for structuring textual technology information based on natural language processing. *Expert Systems with Applications*, 164:114042.

Jawahar, G., Sagot, B., and Seddah, D. (2019). What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Jia, C., Yang, Y., Xia, Y., Chen, Y.-T., Parekh, Z., Pham, H., Le, Q., Sung, Y.-H., Li, Z., and Duerig, T. (2021). Scaling up visual and vision-language representation learning with noisy text supervision. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4904–4916. PMLR.

Jiang, M., Chen, Y., Liu, M., Rosenbloom, S. T., Mani, S., Denny, J. C., and Xu, H. (2011). A study of machine-learning-based approaches to extract clinical entities and their assertions from discharge summaries. *Journal of the American Medical Informatics Association*, 18(5):601–606.

Joung, J. and Kim, K. (2017). Monitoring emerging technologies for technology planning using technical keyword based analysis from patent data. *Technological Forecasting and Social Change*, 114:281–292.

Kao, W.-T. and Lee, H.-y. (2021). Is BERT a cross-disciplinary knowledge learner? a surprising finding of pre-trained models' transferability. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2195–2208, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Karvonen, M. and Kässi, T. (2013). Patent citations as a tool for analysing the early stages of convergence. *Technological Forecasting and Social Change*, 80(6):1094–1107.

Kay, L., Newman, N., Youtie, J., Porter, A. L., and Rafols, I. (2014). Patent overlay mapping: Visualizing technological distance. *Journal of the Association for Information Science and Technology*, 65(12):2432–2443.

Khashabi, D., Chaturvedi, S., Roth, M., Upadhyay, S., and Roth, D. (2018). Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Kim, G. J., Park, S. S., and Jang, D. S. (2015). Technology forecasting using topic-based patent analysis. *JSIR Vol.74(05)*.

Kobayashi, G., Kuribayashi, T., Yokoi, S., and Inui, K. (2020). Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

Kovaleva, O., Kulshreshtha, S., Rogers, A., and Rumshisky, A. (2021). BERT Busters: Outlier Dimensions that Disrupt Transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3392–3405, Online. Association for Computational Linguistics.

Kovaleva, O., Romanov, A., Rogers, A., and Rumshisky, A. (2019). Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, Canadian Institute for Advanced Research.

Krizhevsky, A., Nair, V., and Hinton, G. (2009). Cifar-10 (canadian institute for advanced research). *Technical Report*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Kuhn, J., Younge, K., and Marco, A. (2020). Patent citations reexamined. *The RAND Journal of Economics*, 51(1):109–132.

Kyebambe, M. N., Cheng, G., Huang, Y., He, C., and Zhang, Z. (2017). Forecasting emerging technologies: A supervised learning approach through patent analysis. *Technological Forecasting and Social Change*, 125:236–244.

Lakretz, Y., Kruszewski, G., Desbordes, T., Hupkes, D., Dehaene, S., and Baroni, M. (2019). The emergence of number and syntax units in LSTM language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 11–20, Minneapolis, Minnesota. Association for Computational Linguistics.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020a). Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020b). Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Landis, J. R. and Koch, G. G. (1977). An application of hierarchical kappa-type statistics in the assessment of majority agreement among multiple observers. *Biometrics*, pages 363–374.

Lee, C., Jeon, D., Ahn, J. M., and Kwon, O. (2020a). Navigating a product landscape for technology opportunity analysis: A word2vec approach using an integrated patent-product database. *Technovation*, 96:102140.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2020b). Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Lee, J.-S. and Hsiang, J. (2020). Patent classification by fine-tuning bert language model. *World Patent Information*, 61:101965.

Levesque, H. J., Davis, E., and Morgenstern, L. (2012). The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'12, page 552–561. AAAI Press.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Li, B., Zhu, Z., Thomas, G., Xu, Y., and Rudzicz, F. (2021a). How is BERT surprised? layerwise detection of linguistic anomalies. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4215–4228, Online. Association for Computational Linguistics.

Li, J., Li, D., Savarese, S., and Hoi, S. C. H. (2023). BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. *CoRR*, abs/2301.12597.

Li, S., Hu, J., Cui, Y., and Hu, J. (2018). Deeppatent: patent classification with convolutional neural networks and word embedding. *Scientometrics*, 117(2):721–744.

Li, X., Xie, Q., Jiang, J., Zhou, Y., and Huang, L. (2019). Identifying and monitoring the development trends of emerging technologies using patent analysis and twitter data mining: The case of perovskite solar cell technology. *Technological Forecasting and Social Change*, 146:687–705.

Li, Y., Choudhary, D., Wei, X., Yuan, B., Bhushanam, B., Zhao, T., and Lan, G. (2021b). Frequency-aware SGD for efficient embedding learning with provable benefits. *CoRR*, abs/2110.04844.

Liang, Y., Cao, R., Zheng, J., Ren, J., and Gao, L. (2021). Learning to remove: Towards isotropic pre-trained bert embedding. In *Artificial Neural Networks and Machine Learning – ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part V*, page 448–459, Berlin, Heidelberg. Springer-Verlag.

Lidén, C. and Setréus, E. (2011). Patent prosecution at the european patent office: what is new for life sciences applicants? *Expert Opinion on Therapeutic Patents*, 21(6):813–817.

Linzen, T. and Baroni, M. (2021). Syntactic structure from deep learning. *Annual Review of Linguistics*, 7(1):195–212.

Liu, L., Li, Y., Xiong, Y., and Cavallucci, D. (2020). A new function-based patent knowledge retrieval tool for conceptual design of innovative products. *Computers in Industry*, 115:103154.

Liu, N. F., Gardner, M., Belinkov, Y., Peters, M. E., and Smith, N. A. (2019a). Linguistic Knowledge and Transferability of Contextual Representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Liu, S.-H., Liao, H.-L., Pi, S.-M., and Hu, J.-W. (2011). Development of a patent retrieval and analysis platform–a hybrid approach. *Expert systems with applications*, 38(6):7864–7868.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019b). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Lo, S. L., Chiong, R., and Cornforth, D. (2017). An unsupervised multilingual approach for online social media topic identification. *Expert Systems with Applications*, 81:282–298.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Lu, Y. (2017). Industry 4.0: A survey on technologies, applications and open research issues. *Journal of industrial information integration*, 6:1–10.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Luo, Z., Kulmizev, A., and Mao, X. (2021). Positional Artefacts Propagate Through Masked Language Model Embeddings. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5312–5327, Online. Association for Computational Linguistics.

Magee, C., Basnet, S., Funk, J., and Benson, C. (2016). Quantitative empirical trends in technical performance. *Technological Forecasting and Social Change*, 104:237–246.

Magerman, T., Van Looy, B., and Song, X. (2010). Exploring the feasibility and accuracy of latent semantic analysis based text mining techniques to detect similarity between patent documents and scientific publications. *Scientometrics*, 82(2):289–306.

Maghrebi, M., Abbasi, A., Amiri, S., Monsefi, R., and Harati, A. (2011). A collective and abridged lexical query for delineation of nanotechnology publications. *Scientometrics*, 86(1):15–25.

Màrquez, L. and Rodríguez, H. (1998). Part-of-speech tagging using decision trees. In Nédellec, C. and Rouveirol, C., editors, *Machine Learning: ECML-98*, pages 25–36.

McCoy, R. T., Min, J., and Linzen, T. (2020). BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 217–227, Online. Association for Computational Linguistics.

McCoy, T., Pavlick, E., and Linzen, T. (2019a). Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

McCoy, T., Pavlick, E., and Linzen, T. (2019b). Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Melluso, N., Pardelli, S., Fantoni, G., Chiarello, F., and Bonaccorsi, A. (2021). Detecting bad design and bias from patents. In *Proceedings of the Design Society: International Conference on Engineering Design*, volume 1. Cambridge University Press.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2016). Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Miaschi, A., Brunato, D., Dell'Orletta, F., and Venturi, G. (2020). Linguistic profiling of a neural language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 745–756, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y., editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751.

Mullen, T. and Collier, N. (2004). Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 412–418, Barcelona, Spain. Association for Computational Linguistics.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Niemann, H., Moehrle, M. G., and Frischkorn, J. (2017). Use of a new patent text-mining and visualization method for identifying patenting patterns over time: Concept, method and test application. *Technological Forecasting and Social Change*, 115:210–220.

Nivre, J., De Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., et al. (2016). Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

No, H. J. and Park, Y. (2010). Trajectory patterns of technology fusion: Trend analysis and taxonomical grouping in nanobiotechnology. *Technological Forecasting and Social Change*, 77(1):63–75.

Odat, S., Groza, T., and Hunter, J. (2015). Extracting structured data from publications in the art conservation domain. *Digital Scholarship in the Humanities*, 30(2):225–245.

OECD (2009). *OECD Patent Statistics Manual*. OECD.

Ozcan, S. and Islam, N. (2017). Patent information retrieval: approaching a method and analysing nanotechnology patent collaborations. *Scientometrics*, 111(2):941–970.

Park, H., Kim, K., Choi, S., and Yoon, J. (2013). A patent intelligence system for strategic technology planning. *Expert Systems with Applications*, 40(7):2373–2390.

Park, I., Jeong, Y., Yoon, B., and Mortara, L. (2015). Exploring potential r&d collaboration partners through patent analysis based on bibliographic coupling and latent semantic analysis. *Technology Analysis & Strategic Management*, 27(7):759–781.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. (2018). Image transformer. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064. PMLR.

Pawar, S., Srivastava, R., and Palshikar, G. K. (2012). Automatic gazette creation for named entity recognition and application to resume processing. In *Proceedings of the 5th ACM COMPUTE Conference: Intelligent & scalable system technologies*, pages 1–7.

Pedreschi, D., Giannotti, F., Guidotti, R., Monreale, A., Ruggieri, S., and Turini, F. (2019). Meaningful explanations of black box ai decision systems. In *Proc. of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*.

Pennington, J., Socher, R., and Manning, C. (2014a). Glove: Global vectors for word representation. In *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Pennington, J., Socher, R., and Manning, C. D. (2014b). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Petrov, S., Das, D., and McDonald, R. (2012). A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096, Istanbul, Turkey. European Language Resources Association (ELRA).

Pilehvar, M. T. and Camacho-Collados, J. (2019). WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Pimentel, T., Valvoda, J., Maudslay, R. H., Zmigrod, R., Williams, A., and Cotterell, R. (2020). Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics.

Piskorski, J. and Yangarber, R. (2013). Information extraction: Past, present and future. In *Multi-source, multilingual information extraction and summarization*, pages 23–49. Springer.

Porter, A. L., Garner, J., Carley, S. F., and Newman, N. C. (2019). Emergence scoring to identify frontier r&d topics and key players. *Technological Forecasting and Social Change*, 146:628–643.

Prasanna, S., Rogers, A., and Rumshisky, A. (2020). When BERT Plays the Lottery, All Tickets Are Winning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3208–3229, Online. Association for Computational Linguistics.

Puccetti, G., Chiarello, F., and Fantoni, G. (2021a). A simple and fast method for named entity context extraction from patents. *Expert Systems with Applications*, 184:115570.

Puccetti, G., Giordano, V., Spada, I., Chiarello, F., and Fantoni, G. (2023). Technology identification from patent texts: A novel named entity recognition method. *Technological Forecasting and Social Change*, 186:122160.

Puccetti, G., Miaschi, A., and Dell'Orletta, F. (2021b). How Do BERT Embeddings Organize Linguistic Knowledge? In *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 48–57, Online. Association for Computational Linguistics.

Puccetti, G., Rogers, A., Drozd, A., and Dell'Orletta, F. (2022). Outlier dimensions that disrupt transformers are driven by frequency. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1286–1304, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Qian, P., Qiu, X., and Huang, X. (2016). Analyzing linguistic knowledge in sequential model of sentence. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 826–835, Austin, Texas. Association for Computational Linguistics.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *Technical report*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Rajaee, S. and Pilehvar, M. T. (2021). How Does Fine-tuning Affect the Geometry of Embedding Space: A Case Study on Isotropy. *arXiv preprint arXiv:2109.04740*.

Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016a). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016b). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016c). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Ramanathan, K. (1994). The polytrophic components of manufacturing technology. *Technological Forecasting and Social Change*, 46(3):221–258.

Ranaei, S., Suominen, A., Porter, A., and Carley, S. (2020). Evaluating technological emergence using text analytics: two case technologies and three approaches. *Scientometrics*, 122(1):215–247.

Rao, R., Meier, J., Sercu, T., Ovchinnikov, S., and Rives, A. (2021). Transformer protein language models are unsupervised structure learners. In *International Conference on Learning Representations*.

Regier, D. A., Narrow, W. E., Clarke, D. E., Kraemer, H. C., Kuramoto, S. J., Kuhl, E. A., and Kupfer, D. J. (2013). Dsm-5 field trials in the united states and canada, part ii: test-retest reliability of selected categorical diagnoses. *American journal of psychiatry*, 170(1):59–70.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. In *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD2016, pages 1135–1144.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *Proc. of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 1527–1535.

Righi, C. and Simcoe, T. (2019). Patent examiner specialization. *Research Policy*, 48(1):137–148.

Robinson, D. K., Huang, L., Guo, Y., and Porter, A. L. (2013). Forecasting innovation pathways (fip) for new and emerging science and technologies. *Technological Forecasting and Social Change*, 80(2):267–285.

Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Roller, S., Kiela, D., and Nickel, M. (2018). Hearst patterns revisited: Automatic hypernym detection from large text corpora. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 358–363, Melbourne, Australia. Association for Computational Linguistics.

Rotolo, D., Hicks, D., and Martin, B. R. (2015). What is an emerging technology? *Research Policy*, 44(10):1827–1843.

Rudinger, R., Naradowsky, J., Leonard, B., and Van Durme, B. (2018). Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14, New Orleans, Louisiana. Association for Computational Linguistics.

Salton, G., Fox, E. A., and Wu, H. (1983). Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036.

Sampson, M., Zhang, L., Morrison, A., Barrowman, N. J., Clifford, T. J., Platt, R. W., Klassen, T. P., and Moher, D. (2006). An alternative to the hand searching gold standard: validating methodological search filters using relative recall. *BMC medical research methodology*, 6(1):1–9.

Sanguinetti, M. and Bosco, C. (2015). Parttut: The turin university parallel treebank. In *Harmonization and Development of Resources and Tools for Italian Natural Language Processing within the PARLI Project*, pages 51–69. Springer.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Sanh, V., Wolf, T., and Rush, A. (2020). Movement pruning: Adaptive sparsity by fine-tuning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20378–20389. Curran Associates, Inc.

Sarica, S., Luo, J., and Wood, K. L. (2020). Technet: Technology semantic network based on patent data. *Expert Systems with Applications*, 142:112995.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C. W., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., Schramowski, P., Kundurthy, S. R., Crowson, K., Schmidt, L., Kaczmarczyk, R., and Jitsev, J. (2022). LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Sellam, T., Yadlowsky, S., Tenney, I., Wei, J., Saphra, N., D'Amour, A., Linzen, T., Bastings, J., Turc, I. R., Eisenstein, J., Das, D., and Pavlick, E. (2022). The multiBERTs: BERT reproductions for robustness analysis. In *International Conference on Learning Representations*.

Silveira, N., Dozat, T., De Marneffe, M.-C., Bowman, S. R., Connor, M., Bauer, J., and Manning, C. D. (2014). A gold standard dependency corpus for english. In *LREC*, pages 2897–2904.

Sinha, K., Jia, R., Hupkes, D., Pineau, J., Williams, A., and Kiela, D. (2021). Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2888–2913, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Small, H., Boyack, K. W., and Klavans, R. (2014). Identifying emerging topics in science and technology. *Research policy*, 43(8):1450–1467.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013a). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013b). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Song, B. and Suh, Y. (2019). Identifying convergence fields and technologies for industrial safety: Lda-based network analysis. *Technological forecasting and social change*, 138:115–126.

Song, C. H., Elvers, D., and Leker, J. (2017). Anticipation of converging technology areas—a refined approach for the identification of attractive fields of innovation. *Technological Forecasting and Social Change*, 116:98–115.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., Kluska, A., Lewkowycz, A., Agarwal, A., Power, A., Ray, A., Warstadt, A., Kocurek, A. W., Safaya, A., Tazarv, A., Xiang, A., Parrish, A., Nie, A., Hussain, A., Askell, A., Dsouza, A., Slone, A., Rahane, A., Iyer, A. S., Andreassen, A. J., Madotto, A., Santilli, A., Stuhlmüller, A., Dai, A. M., La, A., Lampinen, A., Zou, A., Jiang, A., Chen, A., Vuong, A., Gupta, A., Gottardi, A., Norelli, A., Venkatesh, A., Gholamidavoodi, A., Tabassum, A., Menezes, A., Kirubarajan, A., Mullokandov, A., Sabharwal, A., Herrick, A., Efrat, A., Erdem, A., Karakaş, A., Roberts, B. R., Loe, B. S., Zoph, B., Bojanowski, B., Özyurt, B., Hedayatnia, B., Neyshabur, B., Inden, B., Stein, B., Ekmekci, B., Lin, B. Y., Howald, B., Orinion, B., Diao, C., Dour, C., Stinson, C., Argueta, C., Ferri, C., Singh, C., Rathkopf, C., Meng, C., Baral, C., Wu, C., Callison-Burch, C., Waites, C., Voigt, C., Manning, C. D., Potts, C., Ramirez, C., Rivera, C. E., Siro, C., Raffel, C., Ashcraft, C., Garbacea, C., Sileo, D., Garrette, D., Hendrycks, D., Kilman, D., Roth, D., Freeman, C. D., Khashabi, D., Levy, D., González, D. M., Perszyk, D., Hernandez, D., Chen, D., Ippolito, D., Gilboa, D., Dohan, D., Drakard, D., Jurgens, D., Datta, D., Ganguli, D., Emelin, D., Kleyko, D., Yuret, D., Chen, D., Tam, D., Hupkes, D., Misra, D., Buzan, D., Mollo, D. C., Yang, D., Lee, D.-H., Schrader, D., Shutova, E., Cubuk, E. D., Segal, E., Hagerman, E., Barnes, E., Donoway, E., Pavlick, E., Rodolà, E., Lam, E., Chu, E., Tang, E., Erdem, E., Chang, E., Chi, E. A., Dyer, E., Jerzak, E., Kim, E., Manyasi, E. E., Zheltonozhskii, E., Xia, F., Siar, F., Martínez-Plumed, F., Happé, F., Chollet, F., Rong, F., Mishra, G., Winata, G. I., de Melo, G., Kruszewski, G., Parascandolo, G., Mariani, G., Wang, G. X., Jaimovitch-Lopez, G.,

Betz, G., Gur-Ari, G., Galijasevic, H., Kim, H., Rashkin, H., Hajishirzi, H., Mehta, H., Bogar, H., Shevlin, H. F. A., Schuetze, H., Yakura, H., Zhang, H., Wong, H. M., Ng, I., Noble, I., Jumelet, J., Geissinger, J., Kernion, J., Hilton, J., Lee, J., Fisac, J. F., Simon, J. B., Koppel, J., Zheng, J., Zou, J., Kocon, J., Thompson, J., Wingfield, J., Kaplan, J., Radom, J., Sohl-Dickstein, J., Phang, J., Wei, J., Yosinski, J., Novikova, J., Bosscher, J., Marsh, J., Kim, J., Taal, J., Engel, J., Alabi, J., Xu, J., Song, J., Tang, J., Waweru, J., Burden, J., Miller, J., Balis, J. U., Batchelder, J., Berant, J., Frohberg, J., Rozen, J., Hernandez-Orallo, J., Boudeman, J., Guerr, J., Jones, J., Tenenbaum, J. B., Rule, J. S., Chua, J., Kanclerz, K., Livescu, K., Krauth, K., Gopalakrishnan, K., Ignatyeva, K., Markert, K., Dhole, K., Gimpel, K., Omondi, K., Mathewson, K. W., Chiafullo, K., Shkaruta, K., Shridhar, K., McDonell, K., Richardson, K., Reynolds, L., Gao, L., Zhang, L., Dugan, L., Qin, L., Contreras-Ochando, L., Morency, L.-P., Moschella, L., Lam, L., Noble, L., Schmidt, L., He, L., Oliveros-Colón, L., Metz, L., Senel, L. K., Bosma, M., Sap, M., Hoeve, M. T., Farooqi, M., Faruqui, M., Mazeika, M., Baturan, M., Marelli, M., Maru, M., Ramirez-Quintana, M. J., Tolkiehn, M., Giulianelli, M., Lewis, M., Potthast, M., Leavitt, M. L., Hagen, M., Schubert, M., Baitemirova, M. O., Arnaud, M., McElrath, M., Yee, M. A., Cohen, M., Gu, M., Ivanitskiy, M., Starritt, M., Strube, M., Swkedrowski, M., Bevilacqua, M., Yasunaga, M., Kale, M., Cain, M., Xu, M., Suzgun, M., Walker, M., Tiwari, M., Bansal, M., Aminnaseri, M., Geva, M., Gheini, M., T, M. V., Peng, N., Chi, N. A., Lee, N., Krakover, N. G.-A., Cameron, N., Roberts, N., Doiron, N., Martinez, N., Nangia, N., Deckers, N., Muennighoff, N., Keskar, N. S., Iyer, N. S., Constant, N., Fiedel, N., Wen, N., Zhang, O., Agha, O., Elbaghdadi, O., Levy, O., Evans, O., Casares, P. A. M., Doshi, P., Fung, P., Liang, P. P., Vicol, P., Alipoormolabashi, P., Liao, P., Liang, P., Chang, P. W., Eckersley, P., Htut, P. M., Hwang, P., Miłkowski, P., Patil, P., Pezeshkpour, P., Oli, P., Mei, Q., Lyu, Q., Chen, Q., Banjade, R., Rudolph, R. E., Gabriel, R., Habacker, R., Risco, R., Millière, R., Garg, R., Barnes, R., Saurous, R. A., Arakawa, R., Raymaekers, R., Frank, R., Sikand, R., Novak, R., Sitelew, R., Bras, R. L., Liu, R., Jacobs, R., Zhang, R., Salakhutdinov, R., Chi, R. A., Lee, S. R., Stovall, R., Teehan, R., Yang, R., Singh, S., Mohammad, S. M., Anand, S., Dillavou, S., Shleifer, S., Wiseman, S., Gruetter, S., Bowman, S. R., Schoenholz, S. S., Han, S., Kwatra, S., Rous, S. A., Ghazarian, S., Ghosh, S., Casey, S., Bischoff, S., Gehrmann, S., Schuster, S., Sadeghi, S., Hamdan, S., Zhou, S., Srivastava, S., Shi, S., Singh, S., Asaadi, S., Gu, S. S., Pachchigar, S., Toshniwal, S., Upadhyay, S., Debnath, S. S., Shakeri, S., Thormeyer, S., Melzi, S., Reddy, S., Makini, S. P., Lee, S.-H., Torene, S., Hatwar, S., Dehaene, S., Divic, S., Ermon, S., Biderman, S., Lin, S., Prasad, S., Piantadosi, S., Shieber, S., Misherghi, S., Kiritchenko, S., Mishra, S., Linzen, T., Schuster, T., Li, T., Yu, T., Ali, T., Hashimoto, T., Wu, T.-L., Desbordes, T., Rothschild, T., Phan, T., Wang, T., Nkinyili, T., Schick, T., Kornev, T., Tunduny, T., Gerstenberg, T., Chang, T., Neeraj, T., Khot, T., Shultz, T., Shaham, U., Misra, V., Demberg, V., Nyamai, V., Raunak, V., Ramasesh, V. V., vinay uday prabhu, Padmakumar, V., Srikumar, V., Fedus, W., Saunders, W., Zhang, W., Vossen, W., Ren, X., Tong, X., Zhao, X., Wu, X., Shen, X., Yaghoobzadeh, Y., Lakretz, Y., Song, Y., Bahri, Y., Choi, Y., Yang, Y., Hao, Y., Chen, Y., Belinkov, Y., Hou, Y., Hou, Y., Bai, Y., Seid, Z., Zhao, Z., Wang, Z., Wang, Z. J., Wang, Z., and Wu, Z. (2023). Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Sternitzke, C. (2010). Knowledge sources, patent protection, and commercialization of pharmaceutical innovations. *Research Policy*, 39(6):810–821.

Suominen, A., Toivanen, H., and Seppänen, M. (2017). Firms' knowledge profiles: Mapping patent data with unsupervised learning. *Technological Forecasting and Social Change*, 115:131–142.

Tay, Y., Dehghani, M., Abnar, S., Chung, H. W., Fedus, W., Rao, J., Narang, S., Tran, V. Q., Yogatama, D., and Metzler, D. (2022a). Scaling laws vs model architectures: How does inductive bias influence scaling?

Tay, Y., Dehghani, M., Rao, J., Fedus, W., Abnar, S., Chung, H. W., Narang, S., Yogatama, D., Vaswani, A., and Metzler, D. (2022b). Scale efficiently: Insights from pretraining and finetuning transformers. In *International Conference on Learning Representations*.

Tenney, I., Das, D., and Pavlick, E. (2019a). BERT rediscovers the classical NLP pipeline. In *Proceedings*

of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Durme, B. V., Bowman, S., Das, D., and Pavlick, E. (2019b). What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Thorleuchter, D., Van den Poel, D., and Prinzie, A. (2010). A compared r&d-based and patent-based cross impact analysis for identifying relationships between technologies. *Technological forecasting and social change*, 77(7):1037–1050.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Timkey, W. and van Schijndel, M. (2021). All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4527–4546, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Torroba Hennigen, L., Williams, A., and Cotterell, R. (2020). Intrinsic probing through dimension selection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 197–216, Online. Association for Computational Linguistics.

Trappey, A. J., Chen, P. P., Trappey, C. V., and Ma, L. (2019). A machine learning approach for solar power technology review and patent evolution analysis. *Applied Sciences*, 9(7):1478.

Tseng, F.-M., Hsieh, C.-H., Peng, Y.-N., and Chu, Y.-W. (2011). Using patent data to analyze trends and the technological strategies of the amorphous silicon thin-film solar cell industry. *Technological forecasting and social change*, 78(2):332–345.

Tseng, Y.-H., Lin, C.-J., and Lin, Y.-I. (2007). Text mining techniques for patent analysis. *Information processing & management*, 43(5):1216–1247.

Turc, I., Chang, M., Lee, K., and Toutanova, K. (2019). Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, abs/1908.08962.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Vicente-Gomila, J., Artacho-Ramírez, M., Ting, M., and Porter, A. (2021). Combining tech mining and semantic triz for technology assessment: Dye-sensitized solar cell as a case. *Technological Forecasting and Social Change*, 169:120826.

Vicente-Gomila, J. M., Palli, A., de la Calle, B., Artacho, M. A., and Jimenez, S. (2017). Discovering shifts in competitive strategies in probiotics, accelerated with techmining. *Scientometrics*, 111(3):1907–1923.

Vig, J. and Belinkov, Y. (2019). Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Volti, R. (2005). *Society and technological change*. Macmillan.

Waight, N. (2014). Technology knowledge: High school science teachers'conceptions of the nature of technology. *International Journal of Science and Mathematics Education*, 12(5):1143–1168.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2019). Superglue: A stickier benchmark for general-purpose language understanding systems. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Wang, J., Yang, Z., Hu, X., Li, L., Lin, K., Gan, Z., Liu, Z., Liu, C., and Wang, L. (2022). GIT: A generative image-to-text transformer for vision and language. *Transactions on Machine Learning Research*.

Warstadt, A., Singh, A., and Bowman, S. R. (2019). Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Wei, J., Garrette, D., Linzen, T., and Pavlick, E. (2021). Frequency effects on syntactic rule learning in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 932–948, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. (2022a). Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Wei, X., Zhang, Y., Zhang, X., Gong, R., Zhang, S., Zhang, Q., Yu, F., and Liu, X. (2022b). Outlier suppression: Pushing the limit of low-bit transformer language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

Williams, A., Nangia, N., and Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Xu, J., Guo, L., Jiang, J., Ge, B., and Li, M. (2019). A deep learning methodology for automatic extraction and discovery of technical intelligence. *Technological Forecasting and Social Change*, 146:339–351.

Yang, Z., Dai, Z., Salakhutdinov, R., and Cohen, W. W. (2018). Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019a). Xlnet: Generalized autoregressive pretraining for language understanding. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Yang, Z., Luong, T., Salakhutdinov, R. R., and Le, Q. V. (2019b). Mixtape: Breaking the softmax bottleneck efficiently. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Yoon, B. and Park, Y. (2005). A systematic approach for identifying technology opportunities: Keyword-based morphology analysis. *Technological Forecasting and Social Change*, 72(2):145–160.

Yoon, J. and Kim, K. (2011). Identifying rapidly evolving technological trends for r&d planning using sao-based semantic patent networks. *Scientometrics*, 88(1):213–228.

Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., and Wu, Y. (2022). Coca: Contrastive captioners are image-text foundation models. *Transactions on Machine Learning Research*.

Yu, X. and Zhang, B. (2019). Obtaining advantages from technology revolution: A patent roadmap for competition analysis and strategy planning. *Technological Forecasting and Social Change*, 145:273–283.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. (2020). Big bird: Transformers for longer sequences. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.

Zeldes, A. (2017). The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Zhang, J., Zhao, Y., Saleh, M., and Liu, P. J. (2020a). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Zhang, K. and Bowman, S. (2018). Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 359–361.

Zhang, M., Fan, B., Zhang, N., Wang, W., and Fan, W. (2021). Mining product innovation ideas from online reviews. *Information Processing & Management*, 58(1):102389.

Zhang, S., Liu, X., Liu, J., Gao, J., Duh, K., and Durme, B. V. (2018). Record: Bridging the gap between human and machine commonsense reading comprehension. *CoRR*, abs/1810.12885.

Zhang, T., Wang, Y., Wang, X., Yang, Y., and Ye, Y. (2020b). Constructing fine-grained entity recognition corpora based on clinical records of traditional chinese medicine. *BMC medical informatics and decision making*, 20(1):1–17.

Zhou, Y., Dong, F., Liu, Y., Li, Z., Du, J., and Zhang, L. (2020). Forecasting emerging technologies using data augmentation and deep learning. *Scientometrics*, 123(1):1–29.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

APPENDIX $\mathcal{A}$

Replicability

In this Section I describe in detail all the experiments carried in Chapter 4, together with the code used to make them this should allow for an effective reproduction of the results. All experiments are carried out using a NVidia A6000 with 48 gbyte of memory.

- For the results in Table 4.1 I fine tune *bert-base-uncased* for 4 epochs on each task with a 2.e-5 learning rate and 256 maximum sequence length. I measure the respective metric for each GLUE task (as defined by Wang et al. (2018)) on the validation set. Both models and datasets are loaded through huggingface `https://huggingface.co/`. For the computation with removed outliers, what I do consists in computing the same metric as for the full model after manually setting to 0 the chosen LayerNorm *weight* and *bias* parameters in all layers. A similar procedure is adopted to compute the values in Table 4.2. Fine tuning on the largest datasets within the glue benchmarks (mnli, qnli, qqp), with the hyperparameters described above on average requires approximately 4000 seconds. The remaining datasets among the glue benchmarks are between 10 to 100 times smaller and require a proportionally scaled amount of time.

- The token counts in Figure 4.3 are obtained through Wikipedia and book corpus by directly using a *bert-base-uncased* and *roberta-base* tokenizers on the whole corpus and counting each token occurrence.

- The results in Figure 4.4 are obtained as follows: for each token in the data (as part of an encoded sequence) I compute the hidden states through a *bert-base-ucased* model and pick the hidden state parameter at the outlier index therefore getting a single numerical value for each token. I also associate to each token its frequency in the pre-training corpus and I measure the Pearson correlation coefficient between this two lists of values.

- The results in Figure 4.5 are obtained by setting LayerNorm *weight* and *bias* parameters at the given outlier index for a given layer to 0. For Figure 4.5a this is done for a model fine-tuned on MNLI train set and the accuracy on MNLI matched is measured, for Figure 4.5b this is done to a pre-trained only model by measuring the MLM loss on the wikitext-v2 validation set (the masking probabilities are kept as in the original BERT paper). This process is repeated for each layer in the model.

- The results in Figure 4.7 are obtained as follows: for each sample in the wikitext-v2 validation set (a single sequence containing n tokens), I encode it with *bert-base-uncased*. This provides attention matrices with size $n \times n$ I take the average over the columns, thus getting a single numerical value for each token. As above, for each token I also collect the hidden state value at the outlier dimension, a single numerical value (for each layer) for each token, and finally I measure the correlation between these two values. In particular since for each layer there are 12 heads I compute 12 correlations at each layer.
- The scores in Table 4.3 are obtained as for Table 4.1 on three instances of *bert-medium* architecture pre-trained with different tokenization strategies. The pretraining of this model is performed with 256 max length, 128 batch size and 1.e-4 learning rate.

Experiments were conducted using a private infrastructure, which has a estimated carbon efficiency of 0.37 kgCO_2eq/kWh (average carbon efficiency in Japan, where the machine is based, for the year 2020). Including experiments that were discarded and failed runs, I estimate that a cumulative of 200 hours of computation was performed on hardware of type RTX A6000 (TDP of 300W). Total emissions are estimated to be 22.2 kgCO_2eq.

A.1 Outliers For Each Model

As mentioned in Subsection 4.2.1 the definition of outliers is not entirely formal: while the weights magnitude let me identify a small subset of weight among which one can search for outliers, there is need to fine tune the model on a downstream task to identify which weights are the most harmful. Table A.1 lists the two most damaging outliers for each model I used in the paper.

Model name	Outlier 1	Outlier 2
”bert-base-uncased”	308	381
”roberta-base”	77	588
”multiberts-seed-1”	218	674
”google/vit-095base-patch16-224-in21k”	187	759
”BERT-medium (ours)”	281	378

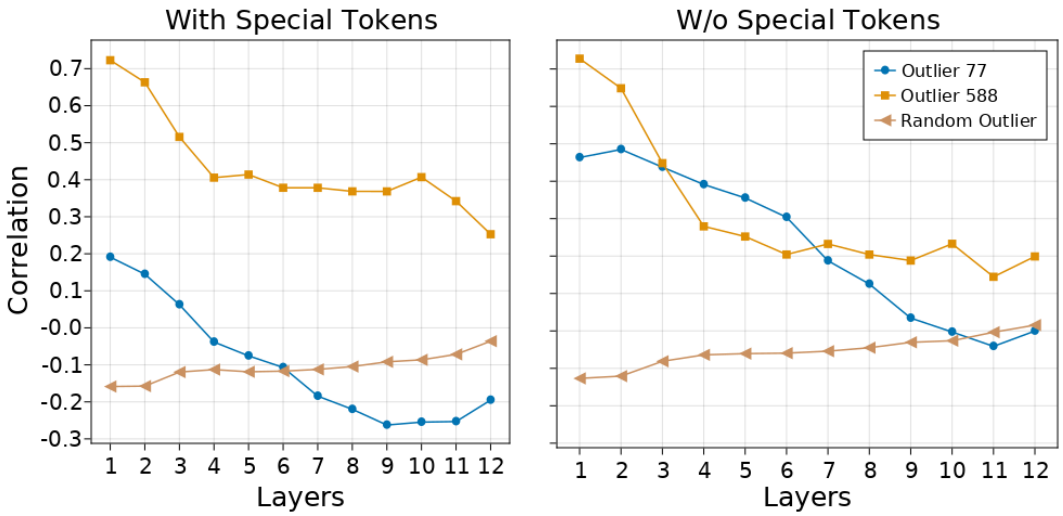
Table A.1: *The outliers identified for each model used in the paper*

A.2 RoBERTa Experiments

The results I showed for BERT-base similarly hold for RoBERTa-base. Table A.2 shows the performance degradation with outliers removed on all GLUE tasks. As shown by Kovaleva et al. (2021) there is one more damaging outlier $O588$ and a less damaging one $O77$, when removed together they cause the largest performance degradation. Figure A.1a and Figure A.2 show that for RoBERTa patterns similar to those I see for BERT in Figure 4.4 and Figure 4.5 appear.

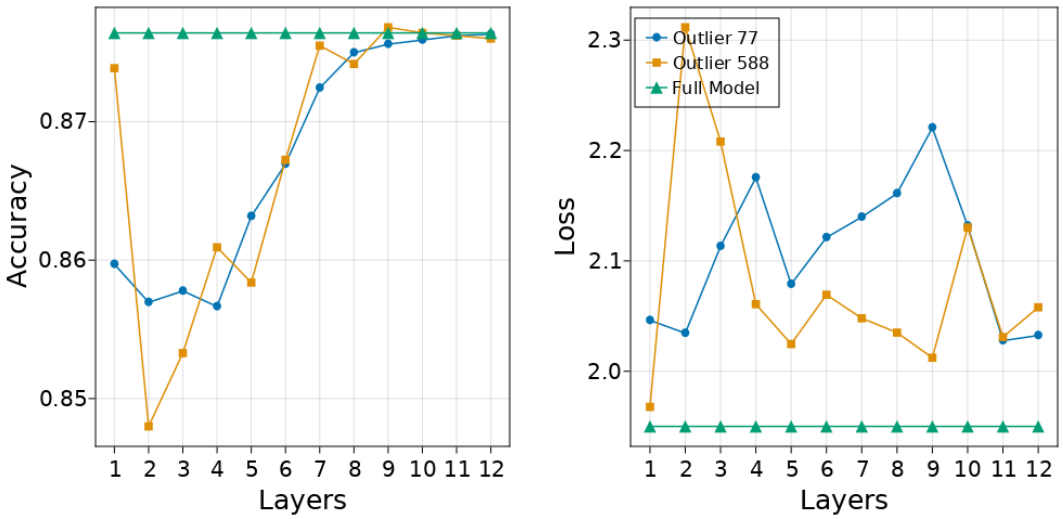
In particular, the two outliers show different behaviour: $O588$ is more damaging when the magnitude of the respective hidden state outlier dimension correlates the most to encoded token frequency in pre-training data. In this case at the earlier layers 2-4 and at layer 10 Figure A.1b I observe a spike in correlation with frequency, and Figure A.2b shows a similar one for MLM loss. On the other hand, $O77$ show the opposite pattern: the less the hidden state dimension corresponding to the outlier correlates to frequency, the more the removal of the LayerNorm outlier damages the model.

For this model I also see an anti-pattern at layer 4 (Figure A.2b): the loss with $O77$ is higher and the one with $O588$ is lower. However, Figure A.1a shows that layer 4 is where the correlation including



(a) *With special tokens* **(b)** *Without special tokens*

Figure A.1: *Correlation of outlier dimension magnitude with token frequency over the Wikitext corpus for a pre-trained RoBERTa-base model. In (a) the correlations accounts for special tokens, in (b) they are excluded.*



(a) *MNLI-m performance* **(b)** *MLM loss (in wikitext-v2)*

Figure A.2: *RoBERTa-base: effect of disabling outliers.*

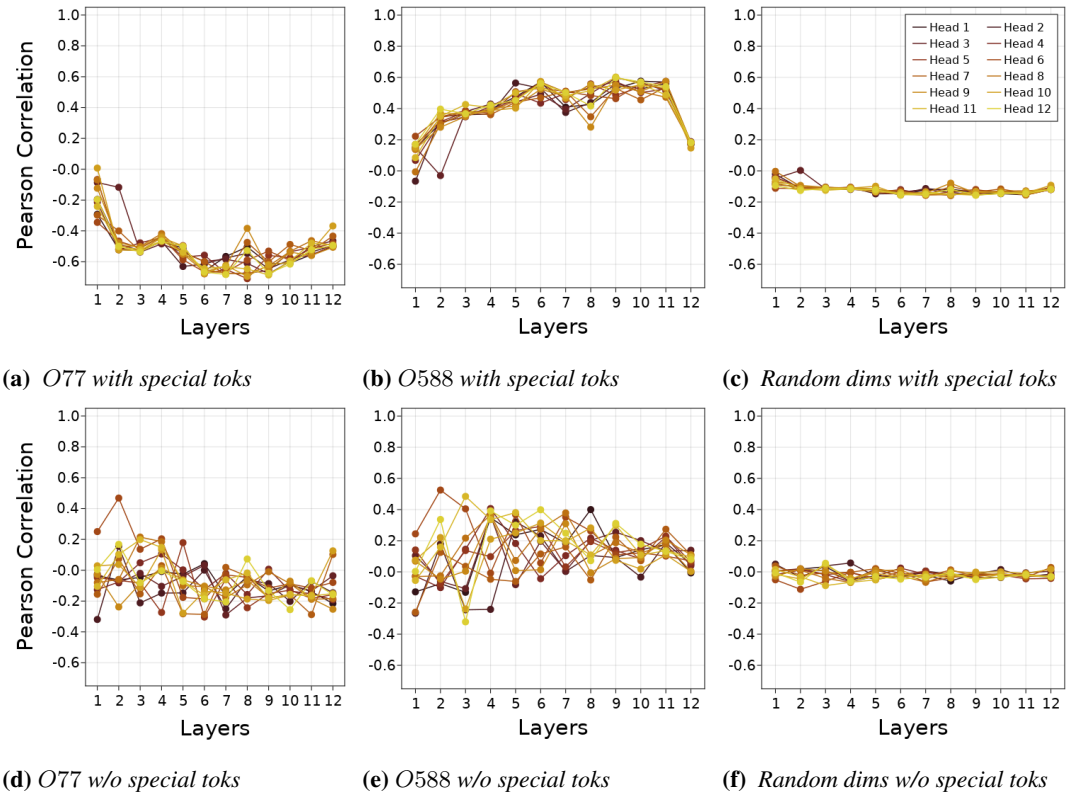
special tokens is closest to zero. One possible explanation for this difference is that RoBERTa pre-training schedule includes a larger number of special tokens Liu et al. (2019b).

Comparing to the BERT experiment I see that although the general pattern is kept, while for BERT the worst layers in term of performance are layers 4-5, for RoBERTa this are layers closer to the input 1-2. One of the reasons behind this difference could be that RoBERTa had longer pre-training.

Finally I also replicate the analysis of attention patterns. Figure A.3 shows for RoBERTa the same patters that Figure 4.7 shows for BERT: for the hidden state parameters corresponding to the outlier dimensions, the correlation values are very different when compared to random ones, both when including the special tokens or not.

Outliers	cola	mnli-mm	mnli	mrpc	qnli	qqp	rte	sst2	stsb
	58.3	87.4	87.6	87.3	92.7	91.4	69.0	95.0	89.1
77	51.5	85.4	85.5	80.1	89.8	90.4	65.0	93.9	83.7
588	7.4	61.5	59.4	70.8	56.6	64.2	54.2	70.3	19.1
588, 77	12.6	45.9	44.9	70.3	50.6	61.2	51.6	68.8	5.4

Table A.2: *Full RoBERTa scores on GLUE benchmarks with outlier effects.*



(a) *O77 with special toks* **(b)** *O588 with special toks* **(c)** *Random dims with special toks*

(d) *O77 w/o special toks* **(e)** *O588 w/o special toks* **(f)** *Random dims w/o special toks*

Figure A.3: *Each figure shows the correlation between the* average query values *in RoBERTa-base self-attention heads, and the magnitude of hidden state parameters at the dimensions corresponding to outlier dimensions. The correlation is computed over examples from Wikitext-v2. Figures (c) and (f) show the average over 10 random dimensions.*

A.3 Outliers in Pre-Training

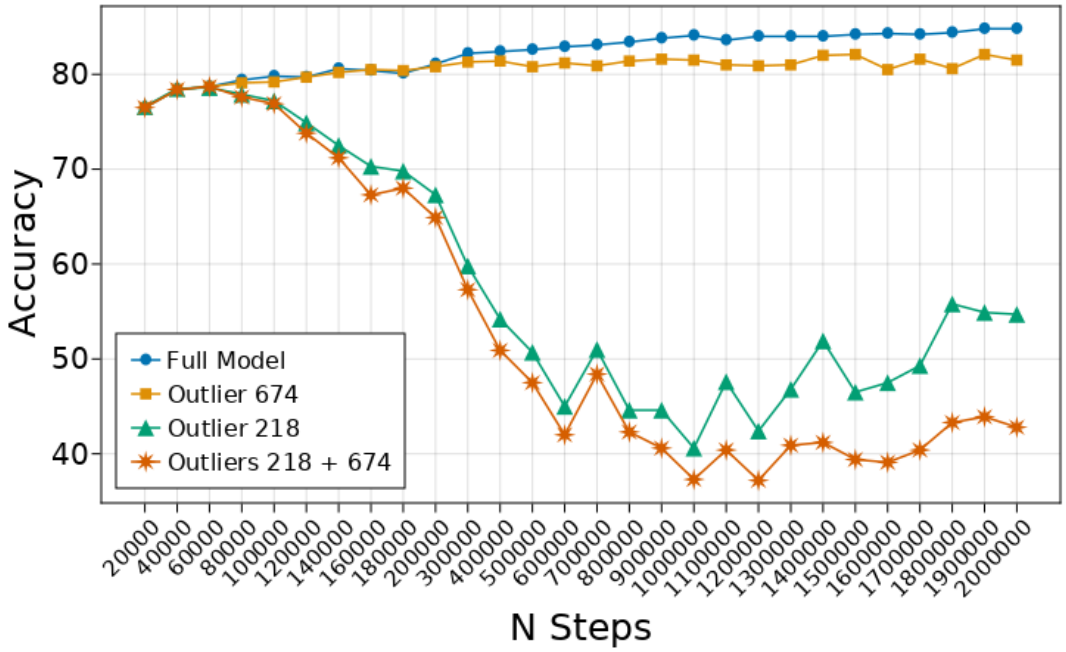


Figure A.4: *The accuracy on MNLI-mismatched of the checkpoints for BERT-base (seed 1), provided by Sellam et al. (2022).*

Figure A.4 and Figure 4.2 show the accuracy on MNLI-mismatched and MNLI-matched respectively, at various checkpoints for BERT-base seed 1 provided Sellam et al. (2022). The results are very similar: early degradation around 80,000 steps, steadily worsening until step 600,000, and then fluctuating further on. The initialization of the classification layer is not fixed across checkpoints.

In Figure A.5 and Figure A.6 I also replicate the experiments while fixing the classification head seed at initialization. In this case as well the results are very close to those from Figure 4.2 and Figure A.4. Specifically, the fluctuating behaviour appearing after 1 million steps is very distinct in this case as well. It is therefore not caused by changes in different fine-tuning initialization but confirmed to be caused by the number of pre-training steps.

An interesting question for future research on this topic is, what is the influence of longer pre-training on this phenomenon, does it get slowly cancelled? Does adding pre-training data from sources other than Wikipedia, the largest source of data for the models I investigate, make the outliers effect smaller or larger?

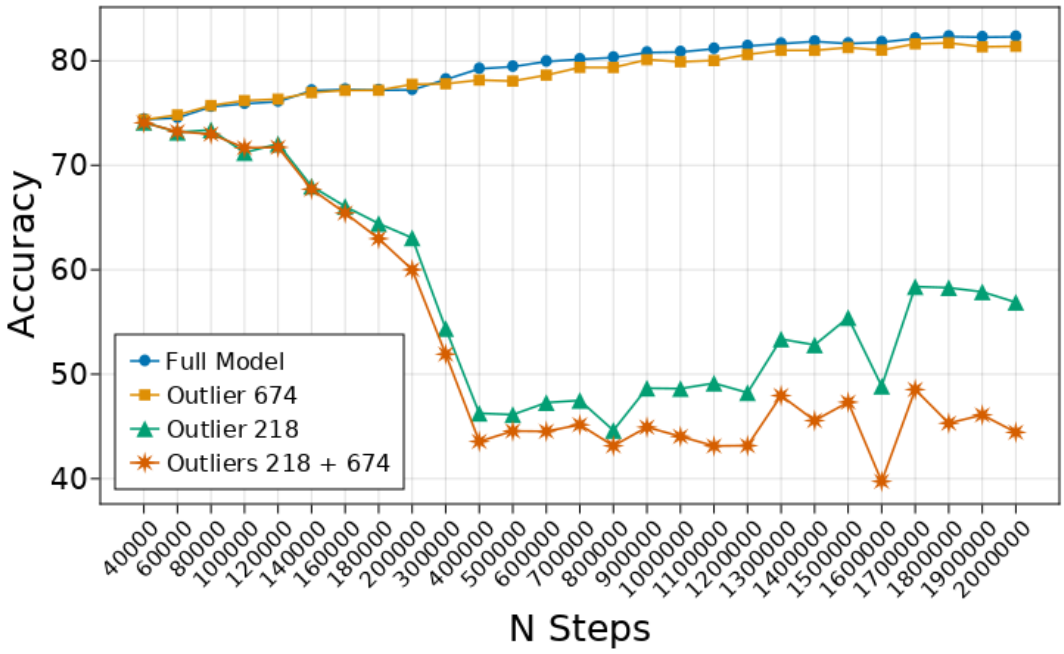


Figure A.5: *The accuracy on MNLI-matched of the checkpoints for BERT-base (seed 1) by Sellam et al. (2022) for full model or with each outlier removed. All classification heads are equally initialized.*

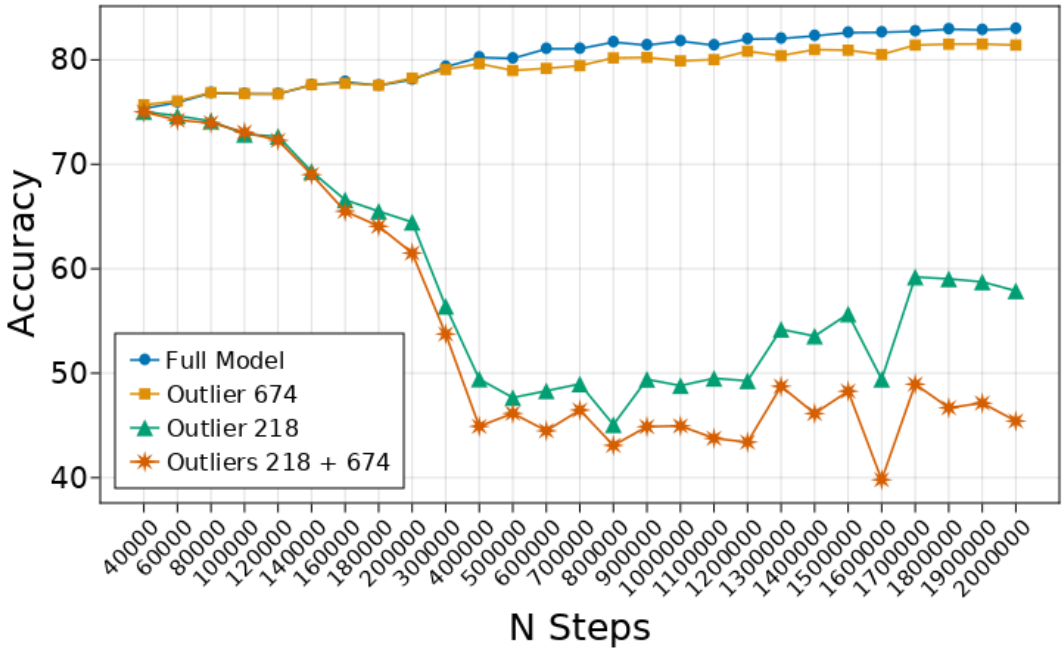
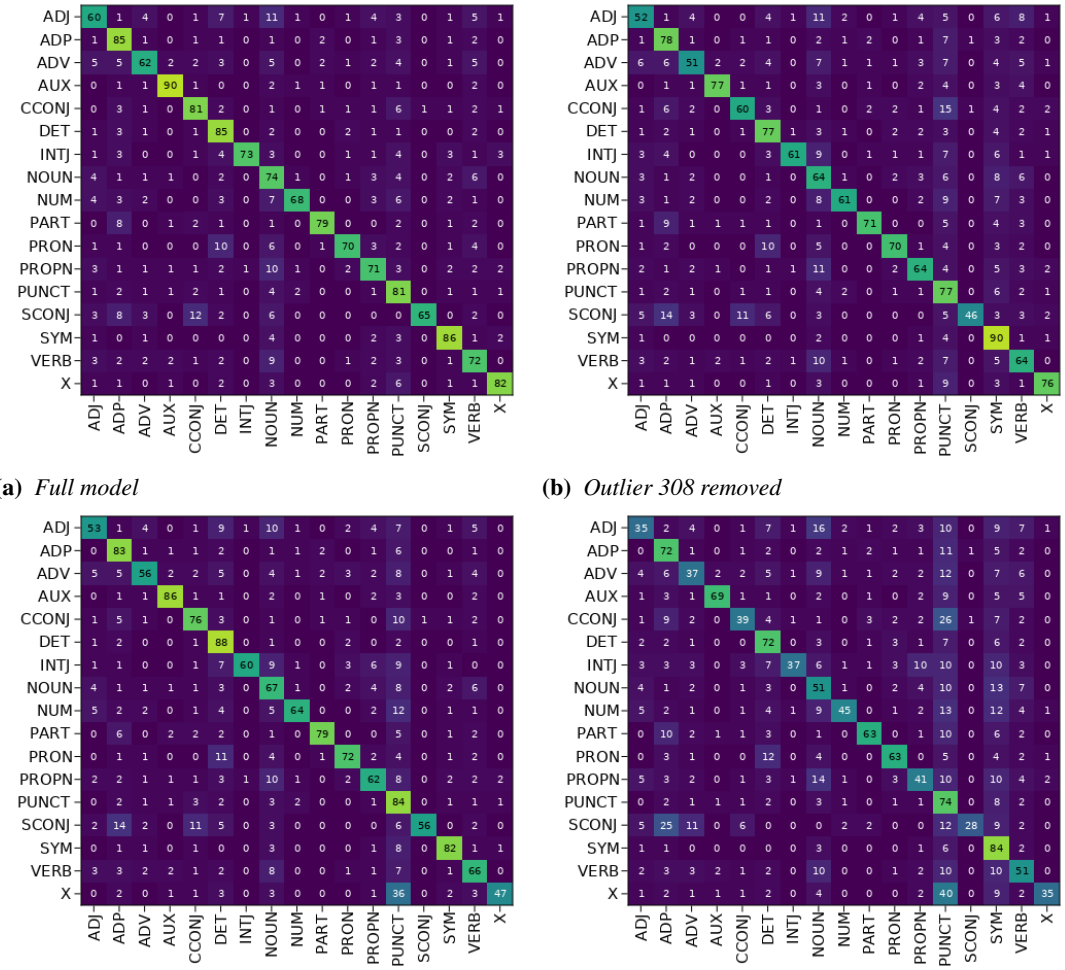


Figure A.6: *The accuracy on MNLI-mismatched of the checkpoints for BERT-base (seed 1), provided by Sellam et al. (2022). All classification heads are equally initialized.*

A.4 POS Tag Distribution of BERT MLM with Disabled Outliers



(a) *Full model*

(b) *Outlier 308 removed*

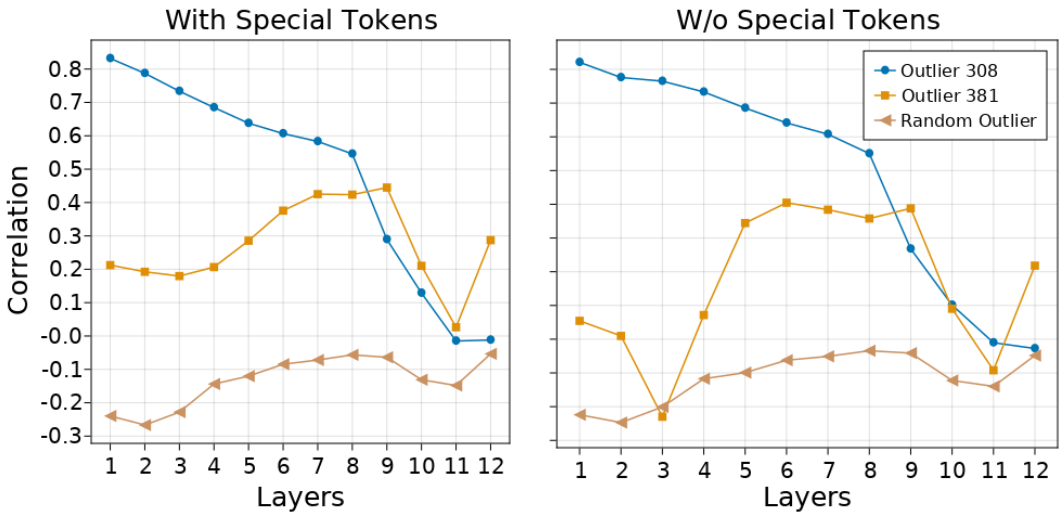
(c) *Outlier 381 removed*

(d) *Outliers 308 and 381 removed*

Figure A.7: *The shift in percentage between the POS tags generated through MLM for full BERT-base model(a) and with different outliers removed, number 308 (b), number 381 (c) and together number 308 an number 381 (d).*

In this experiment I investigated the POS tags of the tokens predicted by the BERT-base MLM with disabled outlier dimensions. Figure A.7 shows the distribution of tags over the replaced tokens. Each row shows the percentage of tags of generated tokens with respect to the tag of the masked token: for example, the top row in Figure A.7d shows that ADJ tokens are replaced with 16% probability by NOUN tokens, with 35% by ADJ tokens, with 10% by PUNCT tokens and so on.

I have previously shown in Table 4.1 that out of two outliers one damages the model performance considerably more. This pattern is also observed here. Figure A.7 shows that individually $O381$ has a much larger effect than $O308$. But now I can also see the qualitative difference between the outliers in the distribution of POS tags of the generated tokens: with only $O381$ disabled, the model becomes more likely to generate nouns and punctuation signs, while $O308$ does not produce so many changes. However, $O308$ has a larger effect in combination with $O381$, again pushing the model towards generating more nouns and punctuation, but also symbols and adpositions.



(a) *With special tokens* **(b)** *Without special tokens*

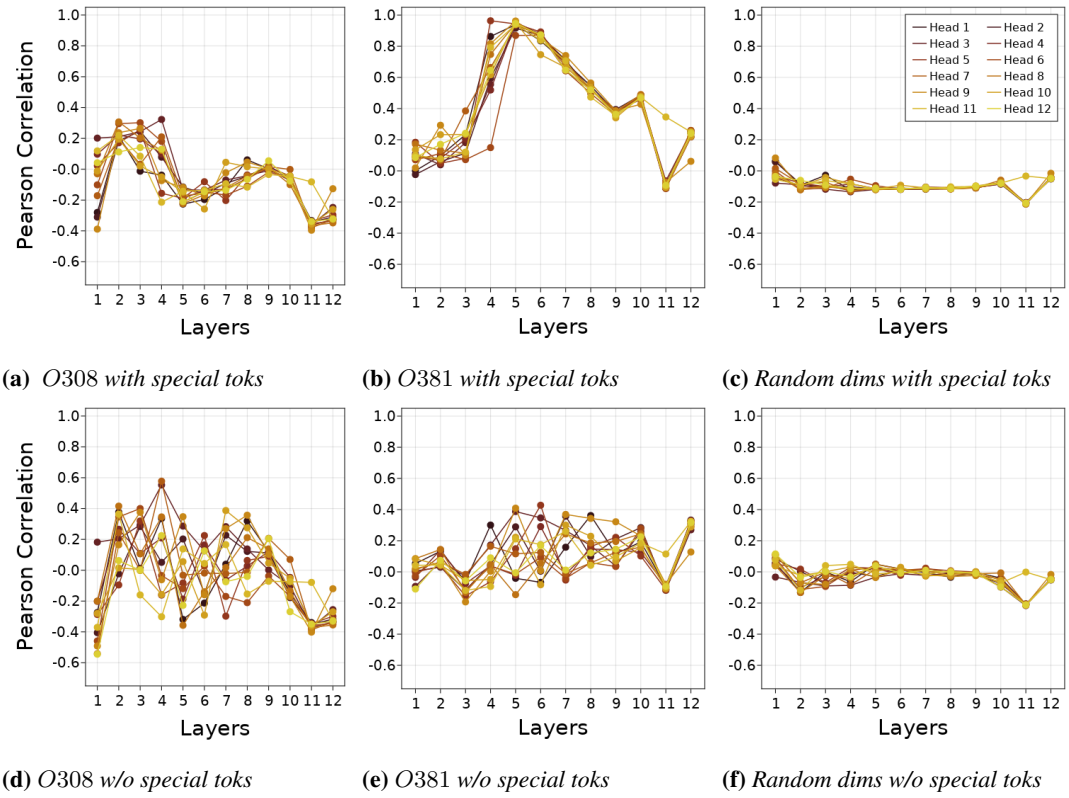
Figure A.8: *BERT-base (fine tuned on MNLI) encoding Wikitext-v2 validation set data: the correlation between magnitude of hidden state parameters corresponding to outlier dimensions, and frequency of encoded tokens in pre-training data.*

A.5 Outliers vs Encoded Token Frequency: the Case of Fine-Tuned Models

To control how much fine-tuning affects the patterns studied, I repeat the experiments with models fine-tuned on MNLI, proceeding as follows: I fine-tune the model using a classification head and then extract the hidden states at each layer and use those in place of the ones of the pre-trained model.

Figure A.8 shows the same information as Figure 4.4, that is the correlation between the hidden states outlier dimension magnitude and the frequency of the encoded tokens in pre-training data, for a BERT-base model fine-tuned on MNLI. The overall patterns is similar to using the pre-trained model, but the correlation values generally decrease: the highest value is now 0.3 when it used to be 0.5 for the pre-trained model. This agrees with the findings from Subsection 4.3.4: the outliers are impacted by the model training.

Investigating attention patterns, Figure A.9 reports the same information as Figure 4.7 for *bert-base-uncased* model fine-tuned on MNLI. In this case I see that the correlations stays high at layers closer to the input data, while those closer to the output have lower values, although in this case as well the values are higher than they are for random outliers Figure A.9c and Figure A.9f.



(a) *O*308 *with special toks* **(b)** *O*381 *with special toks* **(c)** *Random dims with special toks*

(d) *O*308 *w/o special toks* **(e)** *O*381 *w/o special toks* **(f)** *Random dims w/o special toks*

Figure A.9: *Each figure shows the correlation between the* average query values *in a BERT-base fine tuned on MNLI self-attention heads, and the magnitude of hidden state parameters at the dimensions corresponding to outlier dimensions. The correlation is computed over examples from Wikitext-v2. Figures (c) and (f) show the average over 10 random dimensions.*