**LMFingerprints**: Visual Explanations of Language Model Embedding Spaces through Layerwise Contextualization Scores

R. Sevastjanova\(^1\), A. Kalouli\(^2\), C. Beck\(^1\), H. Hauptmann\(^3\), and M. El-Assady\(^4\)

\(^1\)University of Konstanz  \(^2\)CIS - LMU Munich  \(^3\)Utrecht University  \(^4\)ETH AI Center, Zürich

1. **Introduction**

Recently, a wide range of deep-learning language models (e.g., BERT, GPT-2) has been developed, reaching high performance in natural language processing tasks [QSX*20]. These models are pre-trained on large corpora (e.g., Wikipedia articles [DCLT18]), learning language structures in an unsupervised manner. They generate contextualized word embeddings, i.e., word vectors that are specific to the context in which they are used [Eth19]. In this work we refer to context as a token sequence spanning a single sentence. In each model layer, word embeddings capture different characteristics of the context [RKR20]. For instance, it has been shown that syntactic features (e.g., dependency trees) are captured best in BERT’s middle layers [RKR20]. Understanding such differences is crucial, especially for researchers applying language models in their work, e.g., by using embeddings for similarity tasks, fine-tuning models for clas-
The explanation of embedding contextualization is an active research topic, especially in the field of computational linguistics. Most common explainability techniques either use supervised probing methods, i.e., linear classification models predicting specific linguistic properties (e.g., [Eth19]), or apply adversarial testing to conclude about models’ capability of learning specific context properties (e.g., [MPL19]). However, the findings of these two strands of research are often contradictory [SKB*21]. At the same time, visual analytics approaches are used for the explainability of embedding contextualization. Such approaches visualize attention-based mechanisms in transformer-based language models (e.g., exBERT [HSG20]), or explore characteristics of contextualized word embeddings (e.g., [AWLG20]). Despite the existing approaches, there are still open questions regarding the information captured in the embedding vectors. Also, the constant development of new language models dictates the need for a technique that provides a quick overview of the model’s layerwise context specificities.

To contribute to this direction, we present a novel scoring-based technique for explaining embedding contextualization. We propose to quantitatively measure the similarity of tokens to various references (e.g., same tokens in different contexts, tokens within the same context, nearest neighbors) and use them as explanations. The motivation lies in the work by Ethayarajh [Eth19], who introduces scoring functions for embedding explainability. In this paper, we extend this set of scores and group them into two categories: a) the degree of contextualization category contains scores which measure the similarity between embedding vectors of different reference tokens and capture the degree of changes that occur in each embedding layer, and b) the type of contextualization category captures common characteristics of tokens and their nearest neighbors and is measured on token string representations and context properties.

A further, main contribution of this paper is an interactive explanation workspace that visualizes the computed score values. The visual representation of the scores is crucial due to the huge amount of data that is generated and has to be investigated, and because the embedding contextualization differs depending on the token’s role (e.g., meaning or function) in its context [SKB*21]. Visualizations are effective means for generating insights into such (complex) data patterns [KAF*08]. The necessity for a visual representation of score patterns was also motivated by our collaborators from computational linguistics. In their analysis, they typically first spend a vast amount of time to explore numerical patterns and formulate concrete hypotheses, which are then tested in new experiments. A visual representation of the data can speed up this process by providing hints into interesting patterns. The visual solution was designed in an iterative design process in a close collaboration with these experts. To support various analysis tasks, we visualize score patterns in two aggregation levels: (1) A corpus-level score aggregation and visualization provides a quick overview of model-specific contextualization properties; (2) A token-level score aggregation supports more fine-grained analysis on embedding contextualization.

We evaluate the proposed scoring-technique and the visual workspace in two steps. First, we compare score patterns for embeddings extracted from BERT to previous findings by Rogers et al. [RKR20]. Second, we investigate score patterns for six popular transformer-based models and present new insights into model specificities through case studies. The contribution of this paper is three-fold. (1) We introduce a scoring-based technique for explaining word contextualization and group the scores into categories. (2) We integrate these scores into a visual, interactive explanation workspace, which presents layerwise differences between embedding contextualization in multiple language models. We refer to the combination of these scoring and visual approaches as LMFingerprints. (3) We compare score patterns to the related work and provide a broad overview of contextualization differences in six popular language models.

2. Background and Related Work

Research in explaining embedding contextualization has three main directions: probing classifiers, adversarial testing, and visualization.

Probing – Probing tasks aim at unveiling the linguistic properties encoded in neural models. This is achieved through a classification task where the final outputs of the model are used as features to predict a specific linguistic phenomenon [JSS19a]. Most probing experiments have focused on BERT and have shown that the model captures a hierarchy of linguistic information (e.g., [Edm20; JSS19b; LTF19a]): surface linguistic features, such as morphosyntactic information, are captured best in the lower layers, while syntactic properties are best represented in the middle layers. The middle to higher layers capture morphological features best, and semantic information such as word senses and semantic roles, is captured best in the higher layers. This captured hierarchy can also be paralleled to the traditional NLP pipeline of Part-of-Speech (POS) tagging, syntactic parsing, named entity recognition, semantic role labeling and coreference resolution [TDP19a].

Adversarial Testing – Adversarial testing aims at exposing the generalization difficulties of the models, in this way shedding light to their inner workings. Relevant research has shown how such models struggle in capturing basic lexical relations [GSG18], identifying ungrammaticality [ML18], efficiently capturing challenging linguistic phenomena, such as negation [DG*S18; RHMS20] and conditionals [RHMS20], or modeling human reasoning patterns, such as numerical or common-sense reasoning [NRS*18]. These findings, however, seem to contradict the results of the probing studies: if the models are able to capture ’deep’ linguistic information (e.g., about syntactic hierarchies), they should also be able to solve the challenges posed by adversarial test sets.

Contextualization – Despite these contrasting findings, there is consensus that the word embeddings generated by such models are contextualized, i.e., a word has different vector representations across different contexts. Particularly, recent work by [Eth19] shows that, by measuring a word’s contextualization on the basis of self-similarity scores, the embeddings become more contextualized, i.e., more context-specific, in the upper layers of BERT. Additionally, [RYW*19; WRCB19] show that contextualized embeddings generally cluster with one another with respect to word senses.

Visual Embedding Explanations – For explainability purposes, most relevant work has focused on visualizing attentions (e.g., NLIZE [LLL*18], Seq2Seq-Vis [SGB*18], BertVis [Vig19], exBERT [HSG20], SANVis [PNJ*19]) and Attention Flows [DWB20]), and showing how transformers learn. Less work has focused on visualizing word embeddings and showing
what the model learns. The first such tools were applied on static embeddings, such as word2vec, Glove, and fastText, and facilitated analogies [LBT*17] and various other tasks such as local word neighborhoods [BCS19; HG18]. Similarly, the recent approach by [Ber20] explores correlations between embedding clusters in BERT for a single model’s layer at a time. The novelty of our approach is the explanation of contextualized word embeddings through a novel scoring technique, integrated into a visual explanation workspace.

3. Problem Characterization

During our long-term collaboration with computational linguists working with language models, we have identified several requirements for a visual analysis approach supporting the explanation of embedding contextualization. The requirement analysis included several informal interviews with two postdoctoral researchers concerning their typical workflow of using language models in their research. We have also reviewed approaches that are currently used by researchers to explain embedding contextualization, such as probing or adversarial tests, as well as visual approaches (see section 2). In the following, we describe the gathered requirements through Models and Data as well as Users and Tasks [MA14].

Models and Data – To analyze word embeddings, we first need to consider language models that produce them. There are various types of language models, with different architectures, generating word embeddings. For simplicity reasons, in this paper we focus on transformers, which are multi-layer models that use attention mechanisms [VSP*17]. During the training process, each token of the input sequence (in the following, we will refer to the input sequence as the token’s context) gets mapped to a high-dimensional vector using a combination of embeddings that indicate the corresponding token, segment, and position. Transformer models can be of different types (e.g., BERT is an autoencoder and GPT-2 is an autoregressive model) and they can have different learning objectives (e.g., BERT is trained on masked language model and the next sentence prediction tasks, while GPT-2 is trained on the next word prediction task).

It has been shown that language models capture different linguistic properties, i.e., semantics, syntax, surface features [RKR20]. To enable such broad analysis of embedding contextualization, we first need to come up with effective contextualization descriptors (R1). Due to models’ inner-working differences, it is expected that their embedding contextualization differs. Hence, the explanation of the contextualization needs to be generalizable and easily applicable on different language models for an effective model comparison (R2). Depending on the size of the corpus, the number of the extracted embeddings that have to be analyzed can introduce challenges. Previous work [Eth19; SKB*21] has also shown that some tokens (e.g., function words) get contextualized stronger than others. Hence, to get a full picture, it is not sufficient to compute statistical contextualization values on the complete corpus alone, but we need additional visualizations for more fine-grained token-group level analysis (R3). Since different layers of a model capture different types of context information [RKR20], the visual representations must highlight layer-specific contextualization properties (R4).

Users and Tasks – Boggust et al. [BCS19] describe that especially expert users, i.e., data analysts, machine learning experts, and computational linguists analyze embeddings for multiple purposes, among others, for understanding the model’s strength and weaknesses (requires R1, R2, and R3) and the information that it learns in different layers (requires R4). In the following, we showcase several tasks that require prior knowledge on embedding contextualization.

T1: Token Similarity Since language models generate embeddings for each token’s occurrence in the corpus depending on the token’s surrounding contexts, one can use them to disambiguate words with multiple meanings. These embeddings can also be used as fixed features for classification tasks. However, since each layer of the model captures different context’s characteristics, researchers need to decide which layer’s embeddings are the most sufficient for the use case at hand (requires R1, R4).

T2: Model Fine-Tuning Often, language models get fine-tuned on labeled datasets for diverse classification tasks [DIS*20], whereby the initial contextualized word embeddings get adapted to capture task-specific language characteristics. For example, one can freeze layers [LTL19] during the fine-tuning to speed up the training process. Currently, decisions on the layer selection are made through a brute-force approach, i.e., different layers are selected and the model with the highest accuracy is chosen manually [LTL19] or automatically [LAV21], based on the produced accuracy scores. These decisions are currently not informed by the knowledge of what is captured in the embeddings in different layer (requires R1, R4).

T3: Token Importance To evaluate what the model learns during its fine-tuning, one can apply attribution methods [LPK21; ZBRS21], which are computed on the word embeddings adapted during the fine-tuning. In order to avoid making false conclusions (e.g., why a particular group of words such as stopwords has a high or low attribution), one needs to first gain an understanding of the token contextualization, e.g., what type of information is encoded in embeddings, to which tokens words become similar, and what are characteristics that the neighborhoods have in common (requires R1, R3).

4. Contextualization Explanation through Scoring Techniques

In this section, we present the foundation of LMFingerprints – a scoring-based technique for explaining embedding contextualization. Instead of training probing classifiers and analyzing their inputs and outputs, we compute the similarity of different embedding, token, and context characteristics and use them as means of explanation (supports R1). In this section, we present existing scoring techniques and introduce new ones that are computed on the embedding nearest neighborhoods. We categorize these scores into two types (see Figure 2). First, we present scores that describe the degree of contextualization. These scores measure the changes in the embedding vectors themselves, e.g., by applying the cosine similarity function. Second, we introduce new scores that characterize the type of contextualization. These are scores that measure the similarity between nearest neighbors according to different token and context properties. In the following, we present a set of examples for each type. This set of scores is not exhaustive and can be extended to enable testing of further hypotheses.

Data Pre-Processing – The data pre-processing pipeline is shown in Figure 1. For each token-context pair in the corpus, we first extract layerwise embedding vectors. We refer to context as a token sequence spanning a single sentence. For instance, if a token occurs in five sentences and the model has 12 layers, we will extract 5 x 12
embedding vectors for it. If a token occurs multiple times in a single sentence, we extract an embedding for each of its occurrences. Hence, the position of a token in the sentence is stored as its metadata. Next, we extract and store $k$ nearest neighbors for each token embedding. As our previous work shows [KSS*21], the highest similarity is observed between embeddings of the same word used in different contexts. Thus, to explain what information is captured in the embedding vectors, we exclude the same tokens from the token’s nearest neighbors, i.e., we obtain nearest neighbors that are the next most similar ones. The parameter $k$ can vary; we have tested $k=3$ as well as $k=10$ and both produced comparable results. For the score computation as well as visualization purposes, we assign each token to its POS tag. Finally, we compute scores for each token embedding that we introduce in the following section. For analysis purposes, the score values get aggregated. The token-level aggregation represents layerwise average score values for each unique token string. The corpus-level aggregation represents the layerwise average score values for all tokens in the corpus.

4.1. Degree of Contextualization

In the following, we present several example scores that can be used to measure the degree of changes in the embedding vectors. These scores can be computed on different reference tokens (e.g., same token in different contexts or nearest neighbor tokens).

**Token in Different Contexts:** To explain the embedding contextualization, Ethayarajh [Eth19] introduces the self-similarity score, which is “the average cosine similarity of a word with itself across all the contexts in which it appears.” Expectation: The higher the similarity, the lower the degree of contextualization.

**Tokens in the Same Context:** Another score introduced by Ethayarajh [Eth19] is intra-sentence similarity, which is the average cosine similarity between a token’s embedding to other tokens in the same context. Expectation: The higher the similarity, the higher the degree of contextualization.

**Nearest Neighbor Tokens:** The nearest neighbor similarity is computed as the cosine similarity between the token and its nearest neighbor embeddings. Expectation: The higher the similarity, the higher the degree of contextualization. The $k$ diversity score measures the number of unique nearest neighbors (unique strings). Expectation: The larger the diversity, the higher the degree of contextualization.

**Baseline Embedding:** One can also measure the similarity to a baseline embedding. For instance, we can extract embeddings from a token without its surrounding context or from the 0th layer [Eth19] and use them as a baseline. We call them similarity to context-size 0 and similarity to layer 0, respectively. Expectation: The lower the similarity, the higher the contextualization.

4.2. Type of Contextualization

The previous scores measure solely the degree of embedding changes according to diverse reference tokens. To obtain characteristics that are encoded in embedding vectors, we introduce further scores that are computed on token nearest neighbors for different token and context properties. There are many potentially interesting scores that can be designed for measuring the type of contextualization. In the following, we specify example scores covering the three linguistic analysis directions, i.e., semantics, syntax, and surface features. In addition, we describe scores related to specific context properties (e.g., token’s position in the context).

**Semantic Similarity:** Related work has evaluated which layers capture token semantic information in BERT and the current findings are contradictory [RKR20]. To measure this characteristic, we introduce the semantic similarity score. It measures similarity between a token to its nearest neighbors according to the WordNet’s [Fel98] word similarity function [PPM04]. Example tokens with a high semantic similarity: girl – woman. To analyze the contextualization of named entities, we introduce the same named entity category score. It measures how often the nearest neighbors share the same named entity category. Example: Germany (LOC) – Italy (LOC).

**Syntactic Similarity:** Same POS tag score measures how often a token and its nearest neighbors have the same POS tag. Example: drive (VERB) – walk (VERB). Same previous POS tag measures how often the token before the target token has the same POS tag as the token before the nearest neighbor. Example: sunny (ADJ) day – good (ADJ) weather. Same next POS tag measures how often the target token has the same POS tag as the token after its nearest neighbor. Example: day is (AUX) – weather will (AUX).

**Surface Similarity:** To see whether the model captures a token’s lexical representation (i.e., characters of the string), we introduce the lexical similarity score. It is computed as the inverse of the edit distance between a token and its nearest neighbor tokens, normalized to the longest token in the neighbor pair. Example: drive – driver.

**Context Specificities:** The positional similarity score measures how often the token and its nearest neighbors have the same absolute position in their contexts. Example: This is a sunny... (index 3) – It was a good... (index 3). The in context score captures how often tokens that occur in the same context become nearest neighbors. Example: This is a sunny... – A sunny... Same context shows in how many cases the token and its nearest neighbors are part of the same context. Example: This is a sunny... – This is a sunny... And a more restricted version – part of the n-gram – shows how often the token and its nearest neighbors together create a tri-gram. Example: This is a sunny... – This is a sunny...
Figure 3: In Model Fingerprints, the cells (i.e., circles) represent ranks for each score (i.e., column) that are extracted from their aggregated corpus-level values. A bipolar scale is used to highlight the layers with the highest (large yellow circles) and the lowest (large gray circles) score values. White star icons are added to the top two ranks of both ends of the scale.

5.1. Model Fingerprints View

The goal of this visualization is to explain contextualization specificities for multiple language models simultaneously (R2) and highlight characteristics that are captured in different models’ layers (R4).

Visual Design The design of a model comparison view was straightforward; our experts agreed that a matrix-like visualization would support the layerwise representation of corpus-level score values (shown in Figure 3). The idea was inspired by literature fingerprints by Keim and Oelke [KO07]. The columns in the matrix represent scores; the rows – layers of the particular model. We first rank layers according to their aggregated corpus-level score values, and by default represent each model by its ranks (i.e., rank 1-12 in language models with 12 layers). Each cell in the matrix is visualized as a circle that is scaled to the layer’s rank, whereby a bipolar scale is used to highlight the layers with the highest and the lowest score values (i.e., circles with the largest radius). In addition, we use two qualitative colors to show the max (yellow) and min (gray) ranks accordingly. For better readability, we add a white star icon on circles of the top two ranks of both ends of the bipolar scale. We use star icons since they are commonly used as rating icons in diverse applications.

This representation primarily shows score-wise layer differences, i.e., in which layers a particular score has the highest or lowest values. Since embeddings from all layers are extracted from the same corpus (and, hence, are comparable), such layerwise changes can indicate that the model in the particular layer captures the score’s underlying pattern (e.g., POS tags, named entity categories). However, the fingerprints do not tell us how dominant these patterns are.

Interactions – For a detailed score distribution analysis, users can click on a single score (i.e., column) and a line chart showing aggregated corpus-level score values is displayed on top of each model’s matrix. To ease model comparison for a single score, the
we finally switched to the radial design. One of its main advantages
Visual Design – The design of this view was a more laborious
At the same time, the visualization should show the score values
Despite the known limitations of radial charts (e.g., angle-based
is the compactness of the layout [DLR09]; it has been also judged
users can change the view to a score-based representation, where the
columns are reordered to group scores according to their category
(e.g., degree- or type of contextualization). By hovering over a col-
umn (i.e., a score for one model), all scores of the particular model
get highlighted in other score-based matrices (shown in Figure 4).

5.2. Model and Score Comparison View

Model Fingerprints give an overview of aggregated corpus-level
score differences. However, related work has shown that context-
ualization of different word categories differs (i.e., some tokens,
e.g., function words, get more contextualized than others [Eth19;
SKB*21]). To enable the analysis of such differences, we designed
the Model and Score Comparison view. The purpose of this view
is to provide an overview of contextualization specificities for dif-
ferent token-groups (R3), e.g., proper-nouns, function words, etc.
At the same time, the visualization should show the score values
for different model’s layers (R4), and be relatively compact to en-
able the analysis of at least two models simultaneously, displayed
side-by-side (R2). In other words, our goal was to design a compact
visualization that shows layerwise token-group score patterns.

Visual Design – The design of this view was a more laborious
process than for the Model Fingerprints view. Before deciding on
a radial area chart to display token-based score values, we imple-
mented multiple alternative visualizations, among others, a parallel
coordinates plot [JF15] and a zoomable matrix-based visualization
[BBH*16; FAS*20]. We provided depictions of some design alter-
atives as supplementary material to this paper. The parallel co-
ordinate plots represented each layer of the aggregated token-level
score values as a dimension in the plot, resulting in an over-plotted
visualization. To avoid this pitfall, we turned to matrices, as they
do not suffer from occlusion and line crossings [BBH*16]. How-
ever, the matrix required a large area to be displayed. Our experts
reported finding it challenging to compare score values across lay-
ers, and wished a more compact representation. After several failed
attempts to reduce the size of the matrix and improve its readability,
we finally switched to the radial design. One of its main advantages
is the compactness of the layout [DLR09]; it has been also judged
as a natural and therefore a memorable visualization [BVBB*13].
Despite the known limitations of radial charts (e.g., angle-based
comparisons [WDG*19]), their ability of presenting patterns in a
compact way was judged positively by our experts.

The radial design displays all tokens in the corpus, groups them
according to their POS tag, and describes them through their layer-
wise contextualization score values. First, we arrange unique tokens
in radial fashion (by default, sorted alphabetically). To let the users
understand layerwise embedding changes, we visualize the score
values for all 12 layers simultaneously. A single layer is displayed as
a line that connects the score values for all tokens in the corpus (i.e.,
12 lines for a model with 12 layers). An example is shown in Fig-
ure 5. We color the lines according to a sequential color scale (i.e.,
from purple representing layer 1 to orange representing layer 12).
To facilitate readability, we additionally color the area between two
succeeding layers (shown in Figure 5) and decrease their opacity
to see overlapping layers. The design is similar to a braided graph
visualization, whereby each braid has transparency and thus, the
overlapping layers are visible. We further group tokens according to
their most frequent POS tag to ease group-pattern analysis.

To enable model as well as score comparison, we display two
radial charts at once, one on each side of the screen. The displayed
model(s) are selected by the user in the Model Fingerprints view.
At the bottom of the screen, we provide an overview of score ranks,
but this time they represent the selected model(s). The users can
change the visualized score in the radial chart by clicking on a
score’s representation at the bottom of the screen. An enlarged
version of the clicked layer-rank visualization is displayed between
the two radial charts, as shown in Figure 7.

Interactions – The radial visualization provides a good overview
of score patterns for token-groups, but, when the radial chart has
to display many tokens, the readability of token labels is restricted.
And, although one can see general differences between two radial
charts, it is perceptually difficult to compare single token values.
To
We focus on the first two: the selection of specific token groups and when the users hover over tokens in the radial chart, the linear area (shown in Figure 7). This chart is linked to the radial chart, i.e., visualization – a linear area chart placed between the radial charts values, the users can switch from the layer-rank to token comparison visualization. To ease the comparison of the different layers; XLNET’s highest self-similarity is in upper layers, the largest increase is between layer 7 and 8. To compare the token-level score values, the users can switch from the layer-rank to token comparison visualization.

We conduct a two-step evaluation to evaluate both, the scoring functions and visual representation of the score patterns. First, we compare score patterns of embeddings from BERT to previous findings from the BERTology paper by Rogers et al. [RKR20]. Second, we investigate score patterns for six popular transformer-based models and present several new insights through case studies.

**Corpus** – We use a corpus of 800 unique sentences of the RTE-1 [DGMO5], RTE-2 [BDD*06] and RTE-3 [GMDD07] corpora.
These corpora contain sentence pairs originally intended for Natural Language Inference. They stem from the news domain and thus contain variable content. The pairs are split into single sentences.

**Models** – We compare embedding contextualization in six transformer models: bert-base-uncased, bert-base-multilingual-uncased, roberta-base, gpt2, xlnet-base-cased, xlm-mlm-en-2048. For simplicity reasons, we selected models that have 12 layers in their architectures. From each model, we extract layerwise embedding vectors for each token-context pair using the hugging-face library [WDS*20]. Since each model uses a different tokenizer, the number of extracted embeddings slightly varies, but is around 12’000 tokens for each. We use faiss library [JDJ17] to extract 10 nearest neighbor tokens / same tokens for each unique token embedding. For better comparison, we reduce the dataset to tokens that occur at least 5 times in the corpus. For tokens that are very frequent (e.g., function words), we limit their set to max 100 unique embeddings per layer. Tokens are mapped to their POS tags using spacy’s en_core_web_sm model.

**6.1. Use Case: Comparing RW Findings to BERT Fingerprints**

To show the validity of the LMFingerprints scoring technique, we first compare scoring patterns computed on embeddings extracted from BERT to the findings summarized in the paper by Rogers et al. [RKR20]. We show examples for the different linguistic analysis levels, i.e., semantics, syntax, surface, and context properties. As described in section 4, one can design multiple scoring functions to measure contextualization for one of these categories. Thus, there is no 1:1 mapping between scores and probing classifiers described in [RKR20], but we show that both techniques can be used as alternatives since they capture the presence of the different linguistic properties in the same layers.

**Semantics** – According to Tenney et al. [TDP19b], semantic information is spread across the entire model. In the side figure, we show the comparison of the study results by Tenney et al. (see the left hand side, i.e., the R(related)W(ork)) and our score patterns (see the right hand side, i.e., S(core)P(atterns)). The results of experiments conducted by Jawahar et al. [JSS19a] and Cui et al. [CCWZ20] show that semantic features are captured best in upper layers. Our score patterns show that the basic WordNet similarity is most present in early layers, i.e., most of the nearest neighbors of content words have a high general semantic similarity. However, the line chart of aggregated corpus-level score values show that the feature is indeed present in all layers. Remark: the aggregated corpus-level values are computed on the complete corpus, including function word; thus, the average similarity is lower than it would be for content words alone.

**Syntax** – One of the experiments where the existing experimental findings show contradicting results is related to the POS tagging task. In particular, experimental results by Tenney et al. [TDP19b] show that the basic syntactic information, i.e., POS tags, is learned by BERT in early layers (the highest scores for probing POS tags are achieved in the first two layers). However, the probing experiments by Liu et al. [LGB*19] find that POS-tagging is performed best at the middle layers (layers 6-8). Our scoring patterns show that in embedding vectors, the POS tag information is the most present in early layers, but the highest values are in layers 3 and 4. Our scoring patterns show that in embedding vectors, the POS tag information is the most present in early layers, but the highest values are in layers 3 and 4.

Syntactic information is most prominent in the middle layers of BERT. This has been shown by Liu et al. [LGB*19] and Hewitt and Manning [HM19], whose results indicate that in the layers 6-9 it is possible to reconstruct the syntactic tree depth. Our score function, which measures in which layer embeddings capture simple syntactic patterns such as n-grams with same POS tag structures (e.g., ADJ followed by a NOUN), show the same results as probing classifiers – the values are highest in layers 6-9. This indicates that in these particular layers the embeddings indeed incorporate syntactic structure information.

**Surface** – Surface features are captured best in early layers of BERT. The original study was conducted by Jawahar et al. [JSS19a]. The authors trained probing classifiers to test surface features such as sentence length and presence of words in the sentence. Instead, we investigate the token lexical similarity as an example of a further surface feature. It shows whether the nearest neighbors have similar spelling, e.g., token run is similar to token runs or runner. As the side figure shows, both the surface features evaluated by Jawahar et al. as well as our lexical similarity is most present in early layers of BERT.
We further show the effectiveness of our approach by presenting NUM words (a), sort them ascending to their values in layer 1 (b), and highlight tokens with POS tag (c). Year numbers have the lowest contextualization, i.e., their nearest neighbors remain numbers even in the model’s upper layers, which is different from for other function words.

Context Properties – The linear word order is best captured in the early layers of BERT. The original study was conducted by Lin et al. [LTF19b] who trained probing classifiers to predict, among others, the token index in the sentence. Their results show that especially the first three layers of BERT can predict the token’s index in its context. Our score patterns show that nearest neighbors especially in BERT’s first two layers have the same positional index. It is important to notice though that Lin et al. limited the sentences size for their experiment, and tested only specific indices (i.e., $2 \leq n \leq 9$). We did not limit the sentence length nor the positional indexes for testing.

6.2. Case Studies: Token Similarity and Importance

We further show the effectiveness of our approach by presenting insights related to analysis tasks introduced in section 3. These insights were created collaboratively with two experts (postdoctoral researchers) from computational linguistics. In an informal setting, the experts interacted with the workspace and verbally summarized their observations. Before we present insights that were gained for tasks T1 (Token Similarity) and T3 (Token Importance), we summarize specificities of the six models. Model Fingerprints (a subset of models is shown in Figure 3) show that the contextualization in the six models differ. The most similar are BERT and BERT-Multilingual (i.e., BERT-ML) models. In the early layers, their nearest neighbors have a high lexical and semantic similarity, in middle layers the models capture named entity categories and token n-grams, i.e., tokens are more similar to those that have the same previous or following POS tag. In upper layers, tokens within the context become more similar to each other. ROBERTA’s properties differ. Tokens within the context become similar to each other in middle layers, and the similarity between tokens in similar n-gram structures is highest in upper layers.

Among the six language models, XLM encodes the positional information the strongest. This feature is dominant in the early layers of the model. We can confirm findings by Ethayarajh [Eth19] and Cai et. al [CHBC20] that GPT-2 is highly anisotropic, i.e., its embeddings fall within a narrow cone of the space, leading to high cosine similarities. Especially in layer 12 the cosine similarity between tokens is close to 1 (in Figure 4a).

In comparison to other models, token nearest neighbors in upper layers are less frequently from the same context, but rather tokens having the same POS tag. In particular, in the example of an n-gram NEW YORK CITY, BERT learns that YORK is similar to NEW and city, however GPT-2 learns that YORK is similar to Jersey, Angeles, Francisco. Different to the bi-directional learning of BERT, GPT-2 uses an unidirectional learning; it can only reach the left context of the evaluated token. XLNET model aims to use the strength of the GPT-2-like autoregressive model and at the same time, use the bi-directionality of BERT [GCMA20]. As shown in Figure 4b, in XLNET, tokens within the context are most similar, and the semantic similarity and POS tag similarity is less dominant in the embedding vectors.

T1: Token Similarity

Contextualized word embeddings are often used for word similarity tasks, since their embeddings encode the semantic information better than static embeddings [RKR20]. In all six models, the semantic information is dominant in the extracted embedding vectors, whereby it is most dominant in early layers, decreasing in model upper layers. Figure 10 shows that in BERT, semantic similarity is highest in early layers and drops in upper layers, but the decrease is smaller than in XLNET. In GPT-2, the semantic similarity stays high throughout all layers with the exception in layer 12. Hence, for semantic word similarity tasks, one could use embeddings extracted from early layers of the different models, or from GPT-2 from all layers but layer 12.

It has been shown that BERT struggles with the representation of numbers [WWL*19]. We observe that some named entity categories
When reordering tokens according to their values for opposite type, BERT is often used for fine-tuning. This might indicate that these tokens could potentially be more relevant for the classification tasks than it is currently assumed. To state this assumption with a confidence, further experiments are needed.

**T3: Token Importance** BERT is often used for fine-tuning tasks [DIS*20], i.e., the model is trained on a labeled dataset for a classification task, whereby word embeddings get adapted to capture task specific language characteristics. To understand which tokens are the most important for prediction making, one can apply feature attribution methods [LPK21; ZBRS21]. The attribution scores are calculated on embedding vectors, thus their outcomes are related to the degree of embedding contextualization. Some works argue that if an attribution score puts a high weight on function words, then these explanations are more or less meaningless [WTWS20]. We assume that this assumption might be too overestimated. When we look at the stopword contextualization in BERT (Figure 11), one can see that their embeddings have a low self-similarity, especially in the upper layers. By visually exploring the type of contextualization scores, we can see that in the model's upper layers the neighbors are more often tokens from the same context. Moreover, the nearest neighbors of function words are often content words such as NOUNS or VERBS. When reordering tokens according to their values for opposite type in layer 11, it becomes obvious that especially determiners (DET) become similar to content words. The actual neighbors are displayed in the projection view (see Figure 8 with neighbors for token A). Hence, in the upper layers of BERT (and also in other models), the embeddings of function words not only contain the information about their functionality, but also about the semantic meaning of the sentence. This might indicate that these tokens could potentially be more relevant for the classification tasks than it is currently assumed. To state this assumption with a confidence, further experiments are needed.

**Expert Feedback** – The Model Fingerprints view was judged as simple yet effective visualization that provides a quick overview of model differences and enables spotting interesting models/scores for detailed analysis. Although the Model and Score Comparison view requires some onboarding phase, the experts appreciated the simultaneous exploration of different categories of phenomena and, according to them, the parallels and differences across the categories could be observed easily. The experts also provided suggestions for further token-grouping approaches such as syntactic dependencies and semantic roles that would support new analysis directions.

**6.3. Discussion: Observations and Research Opportunities**  
**LMFingerprints** is a useful technique for the explanation of embedding contextualization. We presented how we can use the explanation workspace to gain insights into model specificities as well as token-group differences. During the design and evaluation process, we discovered several opportunities for future research:

- **Contextualization and Attribution Scores** – To understand function word role for classification tasks, one could explore the relationship between the degree of contextualization of specific tokens and their attribution scores. Since attribution score values are computed on embedding vectors, we would expect to see some relationship between the degree of contextualization and their importance.

- **Interaction Techniques for Radial Layouts** – We show that radial charts can provide a compact overview of score properties. It is also possible to see general pattern differences between two models. However, we also faced challenges to compare single token values for the two models, and hence, we applied several interaction techniques to support it. In particular, we extended the view with a linear chart that could be used as a lens showing the actual values for tokens from the two models. We see a potential to extend this technique, in particular, a more intuitive navigation through the data.

- **More Advanced Token Grouping** – Motivated by the related work [Eth19; SKB*21], we group words according to their POS tag and type (i.e., function and content words). More advanced methods of grouping tokens according to further properties or similarity might support testing of further, more specific hypotheses.

- **Collaborative Pattern Exploration** – During the design phase, we had the opportunity to observe the analysis workflow of experts from computational linguistics. They collaboratively searched for patterns, commented their observations, that led to active discussions about potential hypotheses to test. It would be beneficial if the interface supported such collaborative analysis for geographically dispersed collaborators. Collaborative analysis in the same visualization space and pattern annotation could make the process more effective.

- **Limitations** – The computation of embeddings and their nearest neighbors is time-consuming; thus, they are pre-computed and don’t influence the tool’s performance. There might be scalability issues, though, when working on larger datasets with more unique tokens.

**7. Conclusion**  
In this paper, we present LMFingerprints, a novel scoring-based technique that combines contextualization scores with visual explanations to provide insights into embedding contextualization in transformer-based language models. We show the applicability of the technique by comparing score patterns computed on embeddings from BERT to the findings summarized in the Primer of Bertology paper by Rogers et al. [RKR20]. Our score patterns confirm insights from related work. Moreover, we show that some tokens (e.g., function words) have unexpected contextualization, which indicates that we might rethink their role in natural language processing applications. A demo is a part of the LingVis framework [EJS*19] under: https://lmfingerprints.lingvis.io.
References


