IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 17 August 2022, accepted 6 September 2022, date of publication 12 September 2022, date of current version 19 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3205719

SURVEY

A Survey of Text Representation Methods and Their Genealogy

PHILIPP SIEBERS[1], CHRISTIAN JANIESCH[2], AND PATRICK ZSCHECH[3]
[1]Faculty of Business and Economics, Technische Universität Dresden, 01067 Dresden, Germany
[2]Department of Computer Science, TU Dortmund University, 44227 Dortmund, Germany
[3]School of Business, Economics and Society, Friedrich-Alexander-Universität Erlangen-Nürnberg, 91054 Erlangen, Germany

Corresponding author: Christian Janiesch (christian.janiesch@tu-dortmund.de)

This work was supported in part by the Federal Ministry of Education and Research (BMBF) within the Project "White-Box-AI" under Grant 01IS22080; and in part by the Project Management Agency Deutsches Zentrum für Luft- und Raumfahrt e. V. (DLR), DLR-Projektträger.

ABSTRACT In recent years, with the advent of highly scalable artificial-neural-network-based text representation methods the field of natural language processing has seen unprecedented growth and sophistication. It has become possible to distill complex linguistic information of text into multidimensional dense numeric vectors with the use of the distributional hypothesis. As a consequence, text representation methods have been evolving at such a quick pace that the research community is struggling to retain knowledge of the methods and their interrelations. We contribute threefold to this lack of compilation, composition, and systematization by providing a survey of current approaches, by arranging them in a genealogy, and by conceptualizing a taxonomy of text representation methods to examine and explain the state-of-the-art. Our research is a valuable guide and reference for artificial intelligence researchers and practitioners interested in natural language processing applications such as recommender systems, chatbots, and sentiment analysis.

INDEX TERMS Artificial neural networks, genealogy, natural language processing, survey, taxonomy, text representation.

I. INTRODUCTION

Computational understanding of natural language is referred to as a particularly hard problem of science [1] and sometimes it is even described as being "simply too hard" [2]. Nonetheless, the research field of *natural language processing (NLP)* takes on this challenge to enable machines to fully understand human text and speech [3]. The principal obstacle for a machine in NLP is the symbolic nature of text. Although a machine can process it, the meaning of language goes beyond what is represented [4]. To access the underlying information, traditional approaches compile texts according to grammatical rules or derived distance measures by comparing hand-crafted features and lexical information [5]. However, these methods are characterized by either poor scalability or their inability to capture more intricate linguistic features, that is semantic information. As a consequence, the research field arrived at a boundary and has grown stale in the decades following its conception as a testament to the complexity of the underlying problem [6].

Only recently, the interest in the research field has been reignited by the explosion of available text data on the Internet, paving the way for novel data-driven approaches [1], [7]. Artificial neural networks in particular enabled the distillation of linguistic information beyond the symbolic nature of text by representing words as multidimensional dense numeric vectors according to the distributional hypothesis, thereby encapsulating semantic meaning in a so-called *language model (LM)*.

Owed to this achievement, *text representation (TR)* methods have been evolving at an unprecedented rate and the research community is struggling to keep up in providing an overview of the field. We contribute threefold to this lack of compilation, composition, and systematization.

(1) Compilation. While surveys of TR exist, they are characterized by a low coverage of existing methods, high-level

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Liu.

explanations, and narrow perspectives (see also Section VI). We provide a comprehensive compilation of the current state-of-the-art of TR methods, detail their motivation, and describe how they implement the distributional hypothesis to create linguistically rich embeddings.

(2) Composition. Further, the genealogy of approaches that constitute the state-of-the-art is obfuscated by the rapid development of the field. We provide an annotated genealogy of TR methods. Specifically, we conduct an in-depth analysis from a conceptual and chronological viewpoint. We carve out evolutionary phases, streams, and differentiate the branches *size*, *context*, *efficiency*, and *multi-tasking*. Furthermore, we highlight methods that constitute historic NLP milestones due to their particular architectural design, eventuating in superior downstream task performance.

(3) Systematization. Lastly, as a means to provide conceptual guidance on how to judge and classify current and future TR methods, we extend beyond the schemata employed above and provide a conceptual taxonomy to classify TR methods along the dimensions *architecture*, *vocabulary*, *representation*, *domain dependency*, and *training strategy*. We examine each dimension to systematize the current state-of-the-art.

In summary, our survey is the first to provide a comprehensive and detailed review of the state-of-the-art of TR and to propose a genealogy visualizing the interrelations and dependencies of the identified methods. Our research can be a valuable guide and reference for artificial intelligence researchers interested in NLP applications such as recommender systems, chatbots, and sentiment analysis.

Our research is structured as follows: First, we elaborate the fundamentals of NLP, artificial neural networks, and TR. Next, we introduce our survey methodology based on the hermeneutic framework. Subsequently, we present a comprehensive list of TR methods and analyze their lineage to emphasize significant milestones. Lastly, we abstract from our survey and provide a conceptual taxonomy of TR methods, which we employ to examine the state-of-the-art. We close with an overview of related work as well as a conclusion and an outlook.

II. FUNDAMENTALS

NLP is a field of artificial intelligence that strives for a holistic computational understanding of natural language [8]. This is no trivial task as language is highly variable and ambiguous [4]. In fact, it is categorized as AI-complete, meaning that its resolution requires the ''synthesis of human-level intelligence'' with regard to natural language [9]. Consequently, a machine must be able to understand each component of language, that is its phonology, morphology, syntax, semantics, and pragmatics. This is reflected in a set of specific challenges, called *NLP tasks*, that provide a measure as to a machine's linguistic capabilities. These tasks can generally be solved by various methods. However, in recent years the research field has been dominated by machine learning approaches, in particular *artificial neural networks* that enable deep learning [6]. Importantly, such models cannot operate directly on discrete symbolic inputs. Hence, TR becomes a crucial second dimension of NLP.

A. NATURAL LANGUAGE PROCESSING TASKS

As previously outlined, computational language understanding can be roughly broken down into the *lexical*, *syntactic*, *semantic*, and *pragmatic* analysis of text through NLP tasks. Each linguistic layer therein unites previous layers and introduces a new level of complexity that entails more advanced methodologies. In general, the tasks in the lexical and syntactic layers are intermediate, that is they are not valuable on their own, but rather a means to an end for the resolution of more complex tasks [10]. They deal with the accumulation of relevant insights on the inherent structural aspects of language. The tasks in the subsequent semantic and pragmatic layers can be considered higher level. They aim at understanding the meaning of language and the context it is used in. Note that while we present explicit NLP tasks in every linguistic layer in the following, state-of-the-art models aim to integrate the tasks of all layers implicitly.

Lexical analysis. On the lowest linguistic level, lexical analysis studies the structure of words. This encompasses their orthography, morphology, and, in case of spoken text, phonology. The fundamental task in this layer is the tokenization of text, that is its segmentation into smaller parts. These tokens typically represent words, characters, or concatenations thereof, called n-grams. The result of the tokenization is a vocabulary that constitutes the basis for virtually any other NLP task. Because of its central function, it is important to ensure that the tokenization process yields a high-quality vocabulary. The segmentation of meaningful character n-grams (i.e., subwords) poses one challenge. An often-used algorithmic solution is byte-pair encoding, which greedily includes only the most frequent tokens in a corpus in the vocabulary [11]. Higher-level representations, for example words or sentences, can then be constructed by combining the corresponding subwords. Another challenge is the highly variant nature of text. It can be mitigated by a variety of preprocessing steps such as noise removal and text normalization. However, current approaches are becoming capable of modeling a large part of the intricacies of natural language in their billions of parameters and depend less and less on such modifications [12].

Syntactic analysis. Syntactic analysis expands the linguistic scope from words to sentences to consider grammatical rules. A central task in this layer is *part-of-speech (POS)* tagging. It describes the traversal of a sentence to annotate the grammatical class of its constituent words. Typical POS are noun, verb, and adjective but can be more fine-grained [13]. The knowledge of the POS of a word has several possible applications, for instance, to improve word-sense disambiguation. Moreover, POS can be used in conjunction with data augmentation, for example to replace words with their synonym. In addition, dependency parsing can reveal grammatical dependencies between words and phrases. To that

end, a directed graph is created that holds dependency information, for example subject and object of a sentence. A substantial challenge in this layer is the grammatical ambiguity of words [14]. Often, the same word assumes different POS. Another obstacle are multi-word expressions that do not obey standard grammatical rules [15].

Semantic analysis. Semantic analysis aims at discerning the meaning of words and sentences in a language. The comprehension of a given sentence may require the understanding of preceding and succeeding sentences. Accordingly, the linguistic scope expands to involve relationships not only between words in a sentence, but between sentences themselves. A large variety of NLP tasks can be placed on the semantic layer, including text classification, word sense disambiguation, machine translation, or summarization. For a brief, non-exhaustive survey on this topic refer to Otter *et al.* [6]. The challenges for tasks in this layer can be grouped into two categories: semantic relationships between words and semantic relationships between words and the context they appear in. Instances of the former are synonyms and antonyms; instances of the latter are polysemy and multi-word expressions.

Pragmatic analysis. Lastly, pragmatic analysis aims at uncovering the meaning of language beyond what is expressed literally, distinguishing what is said from what is conveyed or accomplished [16]. The linguistic scope broadens to world knowledge and common sense. Linguistic challenges in this layer are concepts such as hyponymy, hypernymy, and meronomy. Infusing a model with the ability to recognize these implications in the situations in which they are important to the meaning of a text is a hard task, not least because they commonly remain unspoken. Yin *et al.* [17] illustrate that a computational pragmatic understanding of natural language cannot yet be fully achieved and that merely atomistic solutions exist. Typically, they involve handcrafted ontologies, for example WordNet,[1] that maintain profound tree-like structures relating words through important concepts.

B. TEXT REPRESENTATION

TR forms the basis of NLP [18]. It is concerned with the adequate encoding and formatting of natural language so that a machine can solve a NLP task. In order for a machine to derive meaning from text and solve more complex tasks, the unstructured, discrete symbols have to be transformed into a structured representation, i.e., numeric vectors. This process is called embedding. The two predominant embedding approaches are *local* and *distributional representations*.

Local representations. Local representations are necessary for the initial conversion of symbols into vectors. Each vector dimension therein uniquely identifies a token of the vocabulary. However, for large corpora the proportion of unique tokens seen in a given text is usually much smaller than the amount of distinct tokens in the vocabulary, resulting in sparse vectors. This *curse of dimensionality* leads to significant problems. Above all, it hinders a model from discovering relevant signals in the input data because it sees only a fraction of the enormous amount of possible feature combinations at any given time [19]. It is therefore essential to control the dimensionality of the vocabulary with growing corpora. Another implication of the alignment between the dimensions of the vectors with the vocabulary is the inability of local representations to express semantic or syntactic information. This is due to vector dimensions being orthogonal to each other and each individual token exhibiting the same distance to any other token in the vocabulary. As a positive effect, local representations are highly interpretable.

The most prevalent local representation method is *one-hot encoding*. For a given text, each token is individually converted into a binary vector. The vectors are filled with zeroes except for a 1 at the dimension that corresponds to the token's index in the vocabulary. Analogue to symbolic representations, one-hot encodings do not provide information on the similarity of tokens. Therefore, they do not facilitate any kind of meaningful analysis of the underlying texts on their own. However, they enable vector and matrix calculations for discrete symbolic inputs, which constitutes the first step for distributional TR with artificial neural networks. This makes one-hot encodings essential for current NLP models. In comparison, a *count vector*, also known as bag-of-words (BOW), constitutes a local representation that is able to capture similarity information. It operates on a document level. To represent a document in a corpus, the vector representations are created as additive compositions of the one-hot encodings of their constituent tokens. Hence, if the same token appears more than once in a document, the summation of the corresponding one-hot encoding leads to document vectors that reflect frequency information.

Distributional representations. Distributional representations build on top of local embeddings to create vectors enriched with linguistic information and overcome the previously discussed disadvantages. To address sparsity, local representations are projected into a shared multidimensional continuous vector space, thereby decoupling the vocabulary from the vector dimensionality to create dense representations. The dimensionality depends on the method that is used for the creation of the embeddings. For instance, if an artificial neural network is used, the dimensionality of the embeddings is prescribed by the hidden layer dimensionality of the network. In order to weave semantic and other linguistic information into the embeddings, tokens are projected according to their context. The contextualization is based on the distributional hypothesis [20], which states that the meaning of a word is defined by the distribution of its neighboring words, that is co-occurrence information. Concisely put by Firth [21]: ''you shall judge a word by the company it keeps''. For example, the words 'apple' and 'orange' would be likely to appear in the same context and would thus occupy similar positions in the vector space. At the same time, words that appear in different contexts would be moved

[1] https://wordnet.princeton.edu/

further away from each other, for example 'apple' and 'brick'. In conjunction, the vector space implicitly captures analogy relationships [22]. Mikolov *et al.* [23] famously demonstrate that the vector operation *vector("king") - vector("man") + vector("woman")* on trained embeddings results in a vector closest to *vector("queen")*. Eventually, the contextualization of the entire vocabulary according to the corpus produces distributional vector representations that abstract arbitrary and complex linguistic concepts across their dimensions. Unfortunately, this makes an interpretation intricate. More so, as each dimension is involved in multiple concepts at once [24].[2]

C. ARTIFICIAL NEURAL NETWORKS

The application of artificial neural networks for NLP can be divided into the creation of distributional representations, that is the contextualization of textual units, and the resolution of downstream tasks, for example text classification. A wide variety of different models can be employed for either objective (for an introduction to machine learning and deep learning cf. also [25]). Nonetheless, a smaller subset of algorithms dominates TR and NLP today, which we present in the following:

Feed forward neural networks (FFNN). The FFNN is the simplest form of an artificial neural network [26]. The central element of an FFNN is the neuron, which controls the signal flow of the network. The neuron takes input signals, multiplies them with a weight, and adds a bias term. The output of the neuron is determined by passing the resulting value through an activation function. This enables the network to distinguish not linearly separable data [26]. FFNN organize neurons into layers, in which the weights connect the output of all neurons in each layer to the input of the neurons in the following layer. Therefore, each layer projects its input onto an n-dimensional vector space, where n is the number of neurons in that layer. FFNN distinguish an input layer processing the original data representation, one or more hidden layers abstracting the output of the input layer by projecting it into a vector space corresponding to their dimensionality, and the output layer mapping the output of the previous layer to a number of neurons corresponding to the possible output values.

Convolutional neural networks (CNN). FFNN are faced with a fundamental problem when being applied to discrete unstructured data: their fully-connected structure lets the number of parameters explode, impeding the ability of the network to learn relevant input signals [27]. In response, a CNN connects the initial neurons only to parts of the input and models higher order dependencies in subsequent layers. Similar to CNN in computer vision that work on pixels as the atomic representation of images, CNN in the NLP domain use the building blocks of text, that is characters or words [28]. As CNN inherently create representations for the entire input sequence, their unmodified application for NLP is limited to tasks that require this coarse representation level. However, the limitation can be resolved by combining CNN with other model architectures [29].

Recurrent neural networks (RNN). RNN are a broad family of artificial neural networks that specialize in the computation of sequential data [30]. They are characterized by an explicit self-loop connection [31]. This allows the model to deal with varying input lengths. The crucial difference to FFNN lies in the fact that the RNN shares its weight matrices. This enables it to generalize the detection of relevant signals in the input to any position and reduces parameter complexity [31]. The key element of any vanilla RNN is its hidden state. This is where the network implements the self-loop connection [31]. The hidden state of the network thus can be compared to a memory that retains the most important information from the previous and current inputs in a compressed form, where the importance of information depends on the training task. Nonetheless, it is a lossy compression as an input vector of arbitrary length is reduced to a fixed length hidden vector [31]. Although RNN specialize in the computation of sequences, their architecture brings with it some issues. The unfolding of the parameter-shared RNN can lead to unstable gradients during the backwards pass with backpropagation through time and it may result in high computational resource requirements and memory usage.

Hochreiter and Schmidhuber [32] introduce a more sophisticated type of RNN, the *Long Short-Term Memory (LSTM)*, to address the problem of unstable gradients of the vanilla RNN. It introduces a dedicated memory mechanism with the cell state that is decoupled from the hidden state of the network and explicitly controlled by three gates: the forget gate, the input gate, and the output gate. This allows the memory to persist over time while being exposed to less encroaching operations than in the vanilla RNN. A variation of the LSTM is the *Gated Recurrent Unit (GRU)* [33]. It conceptually retains the gated architecture of the LSTM but addresses the problem of high computational resource requirements by reducing tensor operations. Most notably, it drops the cell state of the LSTM and returns to using the networks hidden state as its memory mechanism. Furthermore, the forget and input gate are combined into a single update gate. A second gate, the reset gate, provides the update gate with context for the generation of candidate values. It does so by merging a subset of features of the hidden state with the current input.

Transformer. The Transformer is a sequence-to-sequence model consisting of an encoder and a decoder block. The encoder block is made up of several layers with each layer containing multi-head attention, a parameter-shared FFNN, and normalization. The composition of the decoder block is similar, but inserts a masked multi-head attention layer to the front. The principal aspect of the Transformer is that it relinquishes convolutions and recurrence and instead operates on attention. Attention can be described as a mechanism to highlight the importance of input features for a model in a given

[2]Distributional representation is not synonymous to distributed representation [24]. Rather, it is a strict subset that focuses on contextual semantics [5].

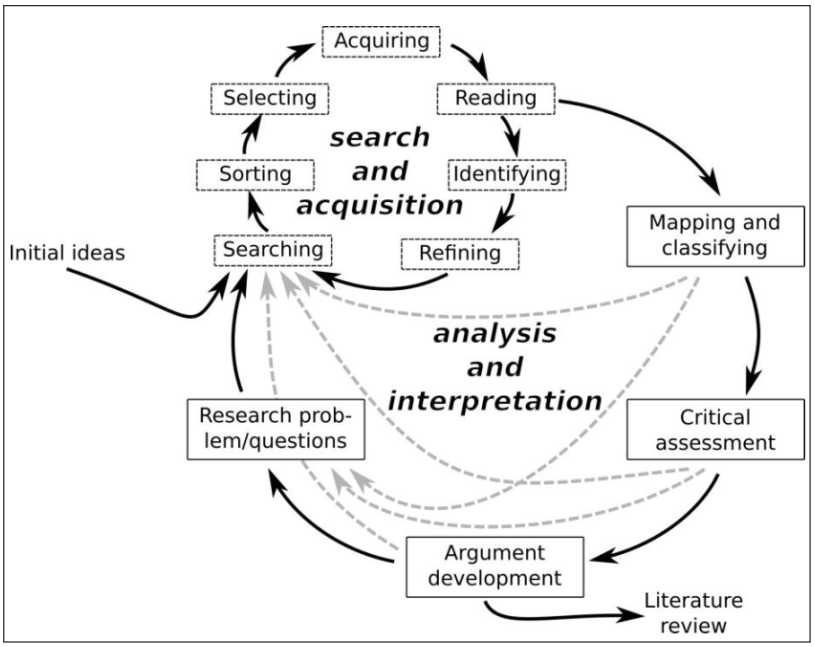


FIGURE 1. Hermeneutic framework of Boell and Cecez-Kecmanovic [38].

task [34]. In the context of TR it can be thought of describing the linguistic composition of tokens in terms of syntactic and semantic similarities to all tokens in a sequence. This includes the influence of a token on itself. As mentioned initially, the Transformer model augments the attention mechanism in two ways. On the one hand, the first layer in the decoder block employs masked attention, which prevents the calculation of attention scores for future tokens in a sequence. This is necessary as the decoder would otherwise be able to indirectly see the ground truth of the current token in a deep setting, preventing any learning effects [35]. On the other hand, all attention scores are calculated with multiple different projections of the input to capture distinct input features. This is called multi-head attention. The Transformer model constitutes a highly parallelizable architecture as it applies attention on the entire input sequence at all layers and shares the FFNN in each layer. Additionally, the Transformer excels in learning long-range dependencies in text by decreasing the path length between the positions of any input and output combination compared to other artificial neural network architectures [36]. However, the attention mechanism is computationally expensive for long sequences [37] and the encoder might be forced to learn irrelevant information of input sequences [30].

III. METHODOLOGY OF LITERATURE REVIEW

We used the hermeneutic framework by Boell and Cecez-Kecmanovic [38] to guide our literature review. The hermeneutic framework construes a literature review as an iterative, interpretative process, in which literature retrieval and literature analysis alternate to facilitate the understanding of a research problem. Figure 1 illustrates the elements that constitute the framework, that is the inner circle of *search and acquisition* and the outer circle of *analysis and interpretation*.

In the *inner circle*, the body of literature is accumulated by identifying relevant publications and refining appropriate search terms. Further, the scholar advances their understanding of the problem domain by reassessing their interpretation of the problem's context and integrating new literature into it. In the *outer circle*, the body of literature is analyzed on a broader scale. Publications are consolidated and compared among themselves with respect to content and methodology. Gaps in the current body of literature are identified to motivate further iterations of the hermeneutic circle. Gaps in the existing research are outlined to build an argument and highlight the research problem. There are several inter- and intra-circle linkages between the activities. They highlight how activities can influence each other while conducting the literature review.

We organized this process within a scope defined according to Cooper's taxonomy [39]: The *focus* is placed on task-agnostic TR method research outcomes of relevant papers, not their underlying theories nor their application or the methodology with which they were derived. The *goal* of the review is the integration of the discovered information so that the reader gets an overview and understanding of recent approaches and their benefits and drawbacks. The *organization* of the review is historical. The findings are presented in a manner that illustrates the evolution of TR along several dimensions. This way it becomes possible to retrace important conceptual changes and challenges, allowing for a comprehensive and descriptive explication of TR over time. The *perspective* is neutral as information is presented objectively. The target *audience* for this literature review are

TABLE 1. Search and acquisition circles of the hermeneutic literature review.

Scope	Findings
First Circle: Overview of NLP	We established a general understanding of NLP before approaching the topic of TR in depth. Following Boell and Cecez-Kecmanovic [38], we queried for surveys and reviews and found that the domain can be organized according to different NLP tasks, which in turn need effective methods (see respective section above). Furthermore, the literature depicted a gradual change from traditional learning to deep learning NLP models.
Second Circle: Overview of TR	We refined our focus to TR. Once more, we started with surveys and reviews. Novel key terms "embedding", "distributional representation", and "distributed representation" manifested. As we found "computational linguistics" and "text mining" to degrade the quality of search results for TR, we excluded those. In the course, different modalities of TR became apparent, of which local and distributional representations were found to be used exclusively in recent literature (see respective section above). Furthermore, we found TR to be shifting towards deep learning similarly to NLP, though at a quicker pace.
Third Circle: TR Methods	We transitioned to searching the blogs for only the most recent influential TR methods. After retrieving the corresponding publications, we conducted a recursive backwards search to retrace the history of TR methods and identify the seminal papers. Our search only uncovered one TR method that is not based on artificial neural networks. After this circle we observed signs of saturation.

specialized scholars as prior knowledge in machine learning and data science is advisable. The *coverage* cannot be exhaustive because of the hermeneutic character of the literature acquisition process. Regardless, a central/pivotal coverage better harmonizes with the historical organization of the review. Additionally, it allows for a more fine-grained implementation of the evolutionary aspect of this paper.

We considered a broad selection of scientific databases for our review, namely SpringerLink, arXiv, Web of Science, ACM Digital Library, IEEE Xplore, and ScienceDirect. We queried title, abstract, and keywords using the high-level term ''natural language processing'', its abbreviation ''NLP'', and the synonymous term ''computational linguistics'' as well as ''text mining'' to include all facets of TR. We excluded the term ''speech'' as speech recognition is a distinct field relying on a different set of methods. We conducted a backward search for each publication to identify additional relevant literature apart from the keyword search. As Boell and Cecez- Kecmanovic [38] point out, a literature search should incorporate sources apart from scientific databases. In response, we included the blogs medium.com and towardsdatascience.com in our literature search.

A *saturation criterion* concludes each hermeneutic circle. Due to its subjectiveness, it is not defined by the framework but manifests as the absence of novel information over the previous iteration. Table 1 summarizes the three search and acquisition circles we conducted.

The literature was retrieved in two steps as shown in Table 2. After the initial search with the respective search terms, we analyzed title and abstract and found 167 potentially relevant publications across all circles. After reading the full text, we discarded 83 publications for a final set of 84 publications.

The initial literature search has been conducted in 2020 and was updated in 2022. The retrieved body of literature roughly covers a time span from 1995 to 2022. The distribution is heavily skewed towards recent publications across all hermeneutic circles. This underlines the radical change TR has undergone, rendering publications earlier than 2013 mostly irrelevant for the current progress in the field.

TABLE 2. Search result of the hermeneutic literature review.

	1st Circle	2nd Circle	3rd Circle	$\sum$
Identified Literature *(Title & Abstract)*	70	41	64	175
Retrieved Literature *(Full text)*	24	16	53	93

This is due to two reasons: First, in that year a paradigm shift in the field of TR has led to the replacement of virtually all previous methods with superior artificial-neural-network-based approaches [6]. Second, this paradigm shift has (re-)ignited interest in the research field, resulting in a substantial increase in publications. In consequence, we only include TR methods from 2013 or later in our review.

IV. EVOLUTION OF TEXT REPRESENTATION METHODS

A. OVERVIEW OF TEXT REPRESENTATION METHODS

In the following, we present and discuss the individual TR methods we identified. Our focus is placed on their motivation and how they implement the distributional hypothesis to create the embeddings. We organized the analysis based on the chronological development of the TR methods, however advancements on a particular method branch are grouped together. Note, that the date of publication is not always coherent with the date of the initial proposal of a TR method. In the following, we use the date of the initial proposal to arrange the TR methods.

word2vec. Mikolov *et al.* [23] propose two architectures, the continuous bag-of-words (CBOW) model and skip-gram model, commonly referred to as word2vec.[3] The most important contribution of these models is the reduction of the computational complexity of calculating distributional representations. This allows the models to scale to large corpora and hence, to produce robust embeddings that accurately capture the linguistic relationships between words.

The *CBOW* model trains the embeddings by sliding a window over the text in the corpus and predicting each

[3] word2vec is the name of the tensorflow implementation.

target word using its preceding and succeeding n neighbours. To that end, a shared weight matrix is used to project the neighbouring words into the same vector space. Thereafter, the sum of the resulting vectors is used to predict the original target word. As a consequence, the vectors of the neighbouring words become aligned with the vector of the target word. However, during the projection of the words, any order information of the tokens is lost, hence the name of the model.

The *skip-gram* model inverts the training process of the CBOW model by using the projection of the target word to predict its n preceding and succeeding neighbours. Because of the higher complexity of the skip-gram training task, Mikolov *et al.* [40] later introduce a number of improvements, most notably negative sampling. When predicting the neighbouring words on the basis of the target center word, this extension provides the model with additional randomly sampled negative words. The model is then trained to both minimize the distances from the target word vector to the vectors of the neighbouring words and maximize the distance to the negative samples.

Global Vectors (GloVe). Pennington *et al.* [41] criticize that previous models like word2vec only account for local word co-occurrence information. With GloVe, they leverage the full statistical potential of corpora by calculating a context-aware global word-to-word co-occurrence matrix and transforming it into a distributed vector representation. Their approach is inspired by artificial-neural-network-based approaches, but it relies on simpler statistics. The co-occurrence matrix is populated at position X_{ij}, if the word w_j appears in the context of w_i, with the context being a window of size n around w_i. Subsequently, the embedding vectors are optimized according to a loss function in such a way, that the dot product of two arbitrary vectors approximate the log probability of the corresponding words in the co-occurrence matrix. To meet this requirement, the dimensions of GloVe embeddings need to capture meaningful information about the global context of a word and about the word itself.

skip-thought. With skip-thought, Kiros *et al.* [42] aim at creating downstream-task-independent and generic sentence representations. For that purpose, the skip-gram model is adapted to reconstruct the preceding and succeeding sentence to an input sentence. It uses a GRU encoder to generate a target sentence representation from the input. During the training of the model, two GRU decoders must predict each word of the preceding and succeeding sentence respectively. For each prediction, they can access the encoder representation of the sentence and the input token at the previous time-step. Finally, the sum of the log-probabilities of the two predicted sentences serves as the loss to condition the encoder representation. A limitation of the skip-thought model is its rather small vocabulary compared to other TR methods, in particular word2vec. To mitigate the problem of *out-of-vocabulary (OOV)* words during inference, skip-thought maps each token of word2vec to the most similar token in the skip-thought vocabulary through linear regression.

Char-CNN. Zhang *et al.* [43] present char-CNN, a CNN that operates on character-level features. They argue that information can be derived from this raw input signal without the necessity for syntactic or semantic knowledge of the underlying language. Furthermore, they question the lack of task specificity of earlier TR methods. Char-CNN is trained on a supervised text classification task. Zhang *et al.* [43] use six convolutional layers. Furthermore, a variety of filter sizes is defined with the intent of capturing different n-gram compositions in the text.

FastText. FastText [44] enriches word embeddings with subword information with the goal of leveraging morphological aspects of words without the need for morphological analysis. The model achieves this through the embedding of character n-grams complementary to the word embeddings. By doing so, non-trivial representations for OOV words are possible by summing the corresponding n-gram vectors. Additionally, complex semantic relationships can be captured through the morphological similarities of words. This aids in the accurate representation of morphologically rich languages, for example the Finnish language. FastText builds on the word2vec model and uses either CBOW or skip-gram with negative sampling. To enable a reliable representation of both rare and OOV words, parameters responsible for the subword embedding are shared. Joulin *et al.* [45] improve the FastText implementation by substantially reducing memory complexity. On the one hand, discriminative pruning retains only the best features, that is words and subwords, under the condition that the entire vocabulary remains covered. On the other hand, embeddings are compressed with a quantization algorithm and hashing is extended from subwords only – as in the original implementation – to subwords and words.

char2vec. Cao and Rei [46] propose the char2vec model with the goal of applying unsupervised morphological analysis on character-level features. The approach is closely related to the FastText model as both models extend word2vec and incorporate the structural information of words. However, FastText does not explicitly aim to uncover morphological aspects of words. Further differences lie in the feature granularity on the one hand and the composition of word representations on the other hand. The learning task of char2vec is similar to the skip-gram model with negative sampling. However, it is adapted by a function that represents the target word as a composite of two half-words. The function employs a *bi-directional LSTM (bi-LSTM)* and a subsequent FFNN with an attention model. The two LSTM read the characters of the target word from left to right and from right to left. Subsequently, the hidden states of both LSTM are concatenated so that each concatenation represents a different split of the target word. With the goal of applying the embedding of the target word to the skip-gram training task, an FFNN is used to reduce the inflated vector dimensionality, resulting from the previous concatenation of two hidden states, back to its original size. An attention model then weighs the importance of each split of the word. Finally, the weighted vectors are

summed and make up the input to the skip-gram task with negative sampling.

context2vec. Melamud *et al.* [47] argue that fixed size context windows, which are used by previous TR methods, inhibit the representation of long range dependencies. Thus, the resulting embeddings reflect only a fraction of the information of a sentence. context2vec [47] constitutes a model for the representation of contexts of variable length to solve this problem. Its architecture is based on the CBOW model with negative sampling but replaces the averaging operation in the projection layer with a bi-LSTM similar to the bi-LSTM used in char2vec but operating on word-level. The resulting vector is a contextualized representation of the words surrounding the target word. It is used to optimize the CBOW learning task with negative sampling.

Very Deep Convolutional Neural Network (VDCNN). The VDCNN [28] highlights the benefit of deeper CNN architectures for NLP, similar to insights gained in the computer vision domain. The model pushes the number of convolutional layers from 6 (see char-CNN) to 29 while continually increasing performance on the jointly trained feature extraction and text classification task. The convolutional layers are organized in blocks as well as batch normalization and use max-pooling operations. Analogue to their computer vision counterparts VGG Net and ResNet, each pooling operation halves the resolution of the feature map and afterwards, the number of feature maps is doubled. The VDCNN uses a constant filter size of three throughout the network. These filters can be thought of to recognize character 3-grams in the first convolutional layer and learned compositions thereof in later convolutional layers. In a deep setting, this makes the approach more flexible than the char-CNN, which prescribes various filter lengths at the first layer.

dict2vec. The premise of dict2vec [48] is that word embeddings can be improved upon using external resources, in particular natural language dictionaries. The definitions in the dictionary entries serve as additional curated contexts for all listed words and provide a mechanism to control the training of the network. Specifically, dict2vec uses the dictionary entries to generate so-called strong and weak word pairs. A strong word pair exists, if a word w_1 is in the definition of another word w_2 and w_2 is also in the definition of w_1. A weak pair exists, if w_1 is in the definition of w_2, but w_2 is not in the definition of w_1. At its base, dict2vec uses the skip-gram model, but extends it with two concepts, positive sampling and controlled negative sampling, that incorporate the discovered word-pair-relationships into the model training. Positive sampling selects a number of words that form weak and strong connections with the target word and introduces a loss so that the dot product of the vectors of the target word and its word pairs is minimized. Controlled negative sampling works similar to negative sampling described by Mikolov *et al.* [40], but substitutes the error-prone random sampling of negative words with the sampling of words that do not form a weak or strong pair. This reduces the probability of taking coincidentally related words as negative examples.

Context Vectors (CoVe). With CoVe, McCann *et al.* [49] draw inspiration from the success of transfer learning in the computer vision domain, where the generalization properties of CNN trained on a vast amount of data are transferred to other task-specific CNN to improve their downstream performance. The authors argue that the abundance of machine translation data can act as a catalyst for NLP models in a similar fashion. CoVe embeddings are created as a complement to previously created word embeddings, in particular GloVe embeddings, to provide them with a dynamic context representation. CoVe embeddings are trained on a supervised sequence-to-sequence machine translation task from English to German. The model architecture consists of an encoder and a decoder. The encoder is a two-layer bi-LSTM that uses word embeddings as its input and generates a sequence of hidden states at the final layer as its output. The decoder is a two-layer uni-directional LSTM that attends over all encoder outputs. In other terms, the final hidden states of the encoder reflect the bi-directional context of every word. Crucially, these representations need to generalize cross-lingual concepts, for example semantics, as they are indispensable for the decoder during the optimization of the machine translation task. Thus, it is the output of the encoder that constitutes the CoVe embeddings.

Universal Language Model Fine-Tuning (ULMFiT). Howard and Ruder [50] agree with McCann *et al.* [49] as to the importance of transfer learning for NLP and recognize more data as a driver of model performance. However, they propose language modeling[4] as the ideal training task instead of machine translation because it entails more data and captures many important linguistic facets. They propose ULMFiT [50], a method that enables efficient transfer learning for any LM. ULMFiT consists of the three training phases (i.e., pre-training, fine-tuning, and classifier fine-tuning) and proposes the three transfer learning principles of discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing.

In the first phase, an arbitrary LM is trained on the vast amount of data the domain offers. In the second phase, the general text embeddings of the LM are adapted to the out-of-distribution downstream task data. During this process, discriminative fine-tuning prevents the model from overfitting to the smaller downstream task dataset by scaling down the learning rate for earlier layers. This preserves the general knowledge about language an LM captures at the first, less task-specific layers. At the same time, slanted triangular learning rates are used for an efficient adaptation of the model. This principle describes a quick initial increase and subsequent gradual decrease of the learning rate of the LM. It ensures that the LM first finds an adequate parameter space and then converges to the local optimum. In the last

[4]In a narrower sense, language modeling is concerned with the generation of the probability of a given sequence of words occurring in a sentence. To generate word probabilities, these LMs are trained on large text corpora and thereby work as a tool to incorporate abundant information in a concise manner that is reusable in an out-of-sample context.

phase, a classifier is fine-tuned. Training all classifier layers simultaneously introduces the risk of over- or underfitting the data. ULMFiT uses gradual unfreezing to solve this issue. First, only the last, that is the least general layer, is trained and then, one by one, additional layers are unfrozen and trained to convergence.

Universal Sentence Encoder (USE). Cer *et al.* [37] conjecture a higher transfer learning potential of sentence-level representations for downstream tasks in comparison to word embeddings. Moreover, they use unsupervised and supervised training to combine the benefits of more data and controlled linguistic training tasks respectively. For the creation of general sentence representations, Cer *et al.* [37] propose the USE. Two encoder variants are considered, a Transformer encoder and a deep averaging network encoder. The former generates contextualized representations for every input token and composes a sentence representation afterwards, while the latter first composes the input embeddings to a sentence embedding and then contextualizes. The variants offer a trade-off between higher accuracy and lower computational complexity respectively.

The encoders are trained by using multi-task learning on a skip-thought task, a supervised conversational input-response task, and a supervised classification task. The conversational input-response task has the model predict the response, given an input. During the classification task, the model is asked to assign a label to the input, that is ''entailment'', ''contradiction'', or ''neutral'', that describes the relationship between a hypothesis and a premise. According to these tasks, the model learns to embed various important aspects of language. During training, the encoder is extended with deep artificial neural networks to form task-specific architectures. However, the parameters of the encoder are shared across all tasks to facilitate the generalization of its sentence embeddings.

Embeddings from Language Models (ELMo). ELMo [29] mitigates the problem of polysemy for feature-based approaches by dynamically generating word embeddings for each input sequence. Furthermore, the method highlights how different layers capture specific linguistic aspects in a deep setting. Peters *et al.* [29] base their conceptual choices on the insights gained by previous TR methods. In particular, they train an LM and use a CNN as a low-level feature extractor. ELMo consists of three layers. The first layer is a character-level CNN that creates word embeddings of the input. These embeddings are contextualized by two layers, which each consist of a bi-LSTM trained on a forward and a backwards language modeling task. In both layers, the hidden states of the two opposite LSTM LMs are concatenated at each index of the input so that they are able to contextualize each word embedding with respect to the entire sequence. The final ELMo vectors are a learned, weighted sum of the output of the three layers of the network. The weighting scheme is downstream-task-specific because the first two layers of the model rather encode syntactic information, while the last layer encodes semantics.

Generative Pre-Training (GPT) With GPT, Radford *et al.* [51] improve upon previous LM-based approaches by completely relying on attention operations [36]. Specifically, the decoder part of the Transformer is used, which implies masked attention and hence an autoregressive LM. In this context, the direct interaction between a token and all previous tokens in a sequence is enabled, enhancing the model's capability to contextualize its embeddings [1]. However, the focus of GPT not so much lies on fine-tuning this setup as on in-context learning and scalability. Radford *et al.* [51] train 12 decoder layers of the Transformer model on a series of downstream tasks with the help of task-specific input transformations and investigate the zero-shot performance of the LM, that is they apply it to a downstream task without any fine-tuning. With *GPT-2*, Radford *et al.* [11] further investigate the zero-shot capabilities of a pre-trained LM to determine, whether fine-tuning limits its expressiveness [52]. To that end, a few changes are made to the original GPT architecture. Most notably, GPT-2 increases the number of layers to a total of 48, leading to a model size that is orders of magnitude larger than any previous NLP model, that is 1.5 billion parameters. *GPT-3* [52] explores the scalability of the GPT model. The depth of a slightly tweaked version of GPT-2 is increased to 96 layers, resulting in 175 billion parameters. Furthermore, Brown *et al.* [52] apply GPT-3 in a few-shot setting, that is conditioning the model to a downstream task by providing a small number of examples.

Bidirectional Encoder Representations from Transformers (BERT). Unlike GPT, Devlin *et al.* [35] only use the encoder block of the Transformer model [36] to train a BERT LM. Because the encoder uses self-attention, each token has direct access to all preceding and succeeding tokens. This allows BERT to train a deeply bi-directional LM. However, the language modeling task of autoregressive LMs such as ELMo or GPT cannot be used. Conditioning these LMs bi-directionally would enable a multi-layer network to trivially predict the target token [35]. Therefore, BERT trains an autoencoder LM, called masked LM, by corrupting the input and then trying to reconstruct it. Precisely, 15% of the tokens in the corpus are statically altered during the data pre-processing step. The alteration process either replaces a token with the [*MASK*] token (80%), a random token (10%) or the original token (10%). All tokens except for [*MASK*] are then used to predict the original tokens. In addition to the masked language modeling task, BERT aims to incorporate knowledge about the relationships between sentences into the model. To that end, two sentences, separated by a special token [*SEP*], form the input to the model. The model subsequently predicts, whether the second sentence follows the first one or is randomly drawn from the corpus. This task is called next sentence prediction. To facilitate a distinction between the tokens in the two segments, a segment encoding is added to the input embedding. With [*CLS*], BERT includes another special token. The token is placed at the beginning of every input sequence and is trained to attend to all tokens

of the sequence. Hence, it is used to capture sequence-level information.

Multi-Task Deep Neural Network (MT-DNN). MT-DNN [53] combines language modeling with multi-task learning before fine-tuning. By adapting the pre-trained BERT LM with several supervised NLP tasks, the model uses a large quantity of cross-task data without overfitting to a specific downstream task, resulting in regularization effects for more effective model fine-tuning. MT-DNN adapts the BERT LM by sequentially feeding input mini-batches of the different tasks to the network and updating the weights of all layers according to the loss function of the respective task. The model thus approximatively reduces the model error on all multi-task objectives simultaneously. MT-DNN uses four task specific layers on top of the LM: single-sentence classification, pairwise text similarity, pairwise text classification, and relevance ranking. During the first task, a label corresponding to the sentiment of a sentence has to be predicted. The second task has the model predict a value that indicates the semantic similarity of two sentences. The third task is similar to the classification task employed in USE. The last task is to rank the best answers to a given query.

Cross-lingual Language Models (XLM). Conneau and Lample [54] criticize the monolingual focus of TR methods and the overrepresentation of the English language. They postulate that multilingual approaches can advance cross-lingual understanding and create high-quality text representations for low-resource languages and propose XLM. The models align token representations across different languages by training on a shared, cross-lingual vocabulary. XLM implements three models each with a different training task: the autoregressive LM of GPT, the autoencoder masked LM of BERT, and a translation LM. The translation LM transfers the masked LM to a supervised setting, in which two parallel sentences are concatenated for the input and aligned with a shared positional-embedding. In addition, a language embedding provides additional information for the model. Subsequently, input tokens are masked and predicted in accordance with the masked language modeling task.

TransformerXL. Dai *et al.* [55] point out two key weaknesses of previous Transformer-based architectures: maximum context distance and context fragmentation. The former limitation restricts the modeling of relationships between tokens to the length of the input because contexts are no longer seen during training. The latter forces the model to train the first tokens in a given input sequence from scratch as context information from previous inputs cannot be accessed. Both limitations therefore result from a limited input sequence length. TransformerXL [55] trains an autoregressive LM that introduces the notion of recurrence to the Transformer architecture to overcome the above mentioned limitations. The outputs of the hidden layers of previous input sequences are cached and concatenated with the hidden states of the current input sequence to predict each token. Once the maximum memory capacity is reached, the oldest cached hidden states are discarded to free up space [56]. In order to enable the recurrence mechanism, the positional encoding of the Transformer, which is statically implemented at the embedding layer, has to be made relative. Otherwise, the same indices would be used for different tokens when attending to previous sequences. Dai *et al.* [55] resolve this issue by extending the attention formula. Specifically, they add trainable biases that adjust for the distance between any two tokens.

Enhanced Representation through kNowledge IntEgration (ERNIE). Sun *et al.* [57] argue that effective text representation methods should use more than just word co-occurrence information of the corpus. ERNIE adapts the masking strategy of the BERT LM by extending it with phrase-masking and entity-masking. Phrases are defined as conceptual units consisting of characters or words. Entities are abstract or concrete concepts that can be denoted with a proper name. In this manner, the model incorporates prior knowledge about language and depends less on long context information. The LM is pre-trained with the help of a three-layer masking strategy. First, 15% of the tokens in the input sequence are masked at random. In the second layer, entities of the input sequences are identified via lexical analysis and the corresponding tokens are masked. Lastly, phrases are identified and masked analogue to the entity masking process. Sun *et al.* [58] later introduced *ERNIE 2.0*, a continuous multitask pre-training framework in line with the insights gained from their previous model proposition ERNIE. They extend the ways in which information other than co-occurence can be leveraged. ERNIE 2.0 enables the pre-training of an LM on different custom training tasks that can be extended by new tasks at any point in time. It pre-trains a BERT encoder on word-aware, structure-aware, and semantic-aware pre-training tasks that improve lexical, syntactic, and semantic capabilities respectively. Word-aware tasks consist of the adapted autoencoder LM introduced by ERNIE, the prediction of whether a token is capitalized and the prediction, whether a token appears at a different position of the same document. Structure-aware tasks are the reordering of a permuted input sequence and a prediction, whether two sentences are adjacent, in the same document or from different documents. Semantic-aware tasks include the prediction of the semantic or rhetorical relatedness of two text spans and a task similar to relevance ranking in the MT-DNN.

MAsked Sequence to Sequence pre-training (MASS). Song *et al.* [59] use a full Transformer to condition a GPT-like decoder on a BERT-like encoder. The encoder of their MASS model is similar to BERT as it masks part of the input. Nonetheless, the employed masking strategy differs. It is tailored to mask spans of text by replacing consecutive tokens with individual mask tokens [M]. The decoder of the MASS model adopts the autoregressive LM of GPT. However, the unmasked tokens for the encoder are masked for the decoder to force the latter to rely more heavily on the input representations of the encoder and thus, the implicit linguistic knowledge encoded in them.

XLNet. Despite their ability to model deep bi-directional contexts, autoencoder LM approaches face two major problems [60]: First, there is a discrepancy between pre-training and fine-tuning. Masked tokens are encountered during pre-training but never in a downstream task. Second, masked tokens are assumed to be independent of each other. They are predicted simultaneously without consideration for other masked tokens of the same sequence. XLNet [60] addresses these problems by creating an autoregressive LM that does not sacrifice bi-directionality, called permuted LM. This is achieved by maximizing the expected log-likelihood of all permutations of an input sequence. Naturally, the permutations include sequences, in which words that appeared after the target word in the original input, now appear before it and vice versa. Because the model parameters are shared the network hence learns bi-directional context information. XLNet permutes its factorization order, not the actual input to not deviate from inputs encountered during fine-tuning. This is achieved with attention masks. However, in a bi-directional context this introduces the problem of trivially predicting the target word [60]. To resolve this issue, XLNet introduces two streams of attention. The content stream, as used in the vanilla Transformer, is a contextualized representation of each input token. The query stream only contains the context representation for each input token. The query stream is used when predicting a token at the current position to hide the token identity and thus prevent a trivial prediction. The content stream is used in all other cases. XLNet accepts the same input structure as BERT, but the network is built on top of a TransformerXL backbone. Furthermore, the concept of relative embeddings introduced by TransformerXL is extended to the segment encoding of BERT, mainly to improve generalization capabilities.

Robustly optimized BERT approach (RoBERTa). Liu *et al.* [61] argue that the original BERT is severely under-trained and that the model can substantially improve in performance through effective and carefully crafted pre-training tasks. To this end, RoBERTa [61] introduces small but effective changes to the vanilla BERT model while leaving the core architecture almost identical. The first key change is making the masked language modeling task dynamic, that is generating the masking pattern each time an input sequence is passed into the network. This increases variance in the training data and thus helps the model to generalize better. The second key change is omitting the next sentence prediction task. This change effectively doubles the context length the model can represent because the input is no longer split into two potentially unrelated sequences. Furthermore, it prevents the introduction of unwanted noise during training [62]. RoBERTa is also trained on more data with larger batches for a longer time and uses a larger byte-pair encoding vocabulary, that is smaller subword units than BERT.

SpanBERT. Joshi *et al.* [62] accentuate the inability of many models to accurately represent multi-word expressions because they optimize the prediction of singular tokens during training. The solution of SpanBERT [62] is straightforward. It adapts BERT to mask spans of natural words in the input. The process is similar to phrase masking in ERNIE, that is each token of a span is replaced with an individual [*MASK*] token. SpanBERT further introduces the span boundary task, in which the model has to recover a complete span using only its boundary tokens. This integrates span-level information into the respective boundary tokens. Lastly, like RoBERTa, SpanBERT omits the next sentence prediction task.

A Lite BERT (ALBERT) The main contribution of ALBERT [63] is its improved parameter efficiency over previous encoder-only Transformer models. Through factorized embedding parametrization, ALBERT disconnects the embedding matrix size from the hidden layer size of the model with a set of smaller matrices. This results in a parameter reduction when the hidden layer size is bigger than the embedding matrix size, allowing efficient scaling of the hidden matrix size according to the modeling needs. Additionally, ALBERT shares all of its parameters across layers by default, further decreasing the amount of model parameters. Although this has a slight negative impact on downstream task performance, it stabilizes the network parameters during training [63]. Moreover, the next sentence prediction task of BERT is not omitted but replaced with a sentence order prediction task. The authors argue that predicting whether a sentence belongs to the same document is too trivial of a task. In their proposed new task, the model has to ascertain whether the order of two consecutive sentences is correct or swapped. Finally, ALBERT implements the n-gram masking strategy of SpanBERT.

Megatron-LM. When increasing the size of an LM, at some point memory limitations pose a substantial problem [64]. Therefore, Shoeybi *et al.* [65] explore the scalability of LMs by training Megatron-LM, which consists of multi-billion parameter versions of BERT and GPT-2. In order to train these models, the authors rely on model parallelism techniques. Specifically, Megatron-LM distributes the calculation of general matrix multiplications across multiple workers. These matrix operations occur in several architectural components of the two previously mentioned models. In FFNN, the matrix multiplication of the weights with the input is split along the weight matrix columns to be distributed on several workers. During the output calculation, the respective weight matrix multiplication is parallelized along its rows. Furthermore, Megatron-LM makes use of the parallel nature of the attention mechanism by splitting attention heads across several workers. However, the output of the attention heads has to be merged. For this, the respective matrix operation is distributed along the rows of the output. Lastly, Megatron-LM lets each worker optimize a set of parameters for layer normalization and dropout themselves to reduce communication requirements between workers.

Building on the Megatron-LM backbone, *Turing Natural Language Generation (T-NLG)* [66] uses advances in distributed computing and distributed memory optimization

technologies (i.e., DeepSpeed and Zer0) to train a Transformer-decoder-only model with 17 billion parameters.

Smith *et al.* [67] go a step further and scale the same Megatron-LM variant to 530 billion parameters. Their model, the *Megatron-Turing Natural Language Generation (MT-NLG)*, is the largest non-sparse model at the time of writing. In order to be able to train a model of such size the authors pair the tensor-slicing of Megatron-LM with a complementary parallelization strategy across three dimensions. First, the model is divided into blocks which are distributed on parallel workers. Second, the individual model layers in each block are allocated to parallel workers. Third, the input data batches are distributed on parallel workers independently of the first two measures. To efficiently coordinate the parallelisms of the model is to minimize the communication overhead between the parallel workers. To this effect, the topology of the hardware cluster is taken into account, that is parallel workers requiring frequent communications are placed in the same or adjacent nodes.

DistilBERT. Similarly to ALBERT, DistilBERT [68] focuses on reducing the computational complexity of a model while maintaining downstream task performance. It uses a setup called knowledge distillation, in which a small student model learns to replicate the behaviour of a larger teacher model. Both teacher and student are versions of BERT, but the network depth of the student is halved. The student is initialized with the weights of the trained teacher and implements three loss functions to align their prediction behavior. A distillation loss aligns the probability distribution over all tokens. The masked language modeling task of BERT is used in a supervised manner to align discrete token predictions, where the ground truth are the predictions of the teacher. Lastly, a cosine embedding loss aligns the embedding vectors for the tokens.

Text-To-Text Transfer Transformer (T5) Raffel *et al.* [69] agglomerate the most valuable insights gained by previous TR models and combine them in the T5, a large-scale text-to-text framework for the resolution of any NLP task. T5 relies on a full Transformer, as encoder- or decoder-only variants would limit downstream task applicability. To distinguish different tasks, the model is fed with a text string that indicates the task that has to be performed, for example ''TL;DR'' for summarization. The initial string is followed by prefixed input-strings in a task-specific structure. As implied by its name, the output of the model is always textual. The model also unifies the objective function to be a maximum likelihood for all tasks. The encoder of T5 uses a span masking task, in which a number of tokens are replaced by a singular mask token. At its base the decoder uses masked self-attention in an autoregressive manner to predict the spans. However, this setup can be limiting in conjunction with prefix information being passed to the model. For instance, in a machine translation task a sentence in the original language is provided so that the model knows what it has to translate. This information would be subject to masking as it forms part of the input. Thus, the model would only be able to access partial information for most time-steps to predict the translation. T5 solves this issue by allowing the decoder to attend to any position of the prefixed input. Beyond that, the attention to previous positions remains prohibited.

Xue *et al.* [70] adapt the T5 model to operate on bytes instead of tokens, eliminating the need for vocabulary generation, text pre-processing, and tokenization. Their model *ByT5* achieves competitive performance when masking longer spans of text and scaling the depth of the encoder.

BART. BART [71] implements the standard Transformer-based neural machine translation architecture. However, the encoder fulfils the role of corrupting the input text according to an arbitrary function and the decoder learns to reconstruct the original in an autoregressive manner. Consequently, BART generalizes the masked LM of BERT and the autoregressive LM of GPT. The approach allows for high flexibility in terms of pre-training input transformations. In particular, BART allows for changes to the length of the input text. This is used to mask spans of text similarly to SpanBERT but with the difference of inserting only a single [*MASK*] token independently of the span length, hence incorporating knowledge about the amount of missing tokens on the decoder side.

Reformer. The Reformer [72] joins the rank of TR methods that aim at reducing computational complexity such as DistilBERT. However, it is the first identified approach that targets the full Transformer architecture. Kitaev *et al.* [72] mainly carve out the attention calculation as a problematic factor in terms of computation. Their solution is the approximation of the key matrix of the attention formula. They reason that only the key values with the highest similarity to the query are required because the attention scores are subject to a softmax function. To sort the keys accordingly, locality-sensitive hashing is used. This hashing function maps each key vector to a bucket and ensures that similar vectors end up in the same bucket. Thus, the attention can be calculated efficiently for each respective bucket. The Reformer furthermore does not materialize the query matrix in memory for memory-efficiency at the cost of performance.

Compressive Transformer. Rae *et al.* [56] propose the Compressive Transformer. The model extends the TransformerXL with a new memory layer to model even longer context dependencies in text. In total, the Compressive Transformer accesses three layers of memory: the current sequence (vanilla Transformer), the TransformerXL memory, and the novel compressed memory. More specifically, instead of discarding old memories like the TransformerXL, a compression function is used to retain lossy representations. The compression function can be max/mean pooling, 1D convolution, dilated convolutions, or most-used, where memories are sorted by their average attention. To facilitate the attention mechanism over previous sequences, the Compressive Transformer adopts the relative positional encoding of the TransformerXL.

ProphetNet. Qi *et al.* [73] postulate that autoregressive LM overvalue local correlations of words at the cost of global

dependencies. To address this issue, they adapt the autoregressive language modeling task to predict n tokens for each token in a sequence. Hence, the model is forced to plan for future tokens. The change is implemented by extending the two-stream attention mechanism of XLNet to n-stream self attention, where each stream i is responsible for the prediction of the future i-th token. ProphetNet further incorporates the masked LM of BERT on the encoder side of its Sequence-to-Sequence (Seq2Seq) architecture.

Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA). ELECTRA [74] is similar to an inverse variant of DistilBERT, that is it uses a small network to optimize a large network. The focus of this approach lies on improving parameter efficiency as well. In particular, the masked LM of BERT is extended with a replaced token prediction task. This novel task is implemented by concatenating two models, a generator and a discriminator. The generator is a reduced version of BERT that trains on the masked language modeling task to generate and pass on plausible token replacements for 15% of the input sequence. Subsequently, the discriminator, a larger BERT, is initialized with the weights of the generator and has to predict for every token, whether it appears in the original sequence or has been altered by the generator. This procedure enables the discriminator to update the weights for all tokens per training step in comparison to only those of masked tokens in the masked language modeling task. Furthermore, no masked tokens are passed to the discriminator. This solves the pre-training/fine-tuning discrepancy of autoencoder LM discussed by XLNet.

MPNet. MPNet [75] leverages the advantages of permutative language modeling of XLNet and the masked LM of BERT while mitigating their respective discrepancies between pre-training and fine-tuning. For this purpose, the authors create a masked and permuted language modeling task. In particular, MPNet can access 100% of the positional information of a sequence in contrast to permutative language modeling and uses bidirectional context information between predicted tokens in contrast to the masked LM. This is implemented by permuting a token sequence and splitting it into three parts: non-masked tokens, mask-tokens, and the tokens that have been masked. Subsequently, self-attention is applied to the first two parts, while two-stream self-attention as in XLNet is applied to the entire token sequence.

Funnel-Transformer. The Funnel-Transformer [76] improves computational efficiency by gradually reducing the hidden vector length with increasing model depth. The respective pooling function reduces the length of the hidden vector representation of the model with a sliding window mean pooling operation. Importantly, the resulting representation is not fed to the next layer directly but is only used to create the query vector for the self attention operation, while the full sequence representation is used for the key and value vectors. This makes the resulting pooled representation more context sensitive. However, due to the dimensionality reduction, the Funnel-Transformer loses the capability of representing individual input tokens. Since this might be required for the resolution of a downstream task, Dai *et al.* [76] propose a decoder to recuperate expressive token-level representations. Concretely, the last hidden representation of the model is first upsampled to the original sequence length by repeating hidden vectors. Then, the last hidden vectors of the uncompressed part of the encoder are added to the upsampled hidden vectors. Finally, the result is passed through several additional Transformer layers to refine the token-level representations.

BigBird. Zaheer *et al.* [77] emphasize the low theoretical understanding of the self-attention operation of the original Transformer. Hence, they question the necessity of full self-attention, which scales quadratically with sequence length, for good NLP performance. BigBird uses sparse matrix calculations in three distinct attention patterns that retain the expressiveness and flexibility of the model while reducing computational complexity to be linear. The first pattern is random attention. Here, each query attends to a random number of randomly chosen keys. The goal is to approximate some characteristics of the full self-attention. The second pattern is called window attention. It aims at capturing local relations between tokens as each token attends to a certain number of preceding and succeeding tokens. The third pattern is global attention, in which a specific added or a chosen existing token attends to and is attended to by every other token in the input. This captures sequence-level information in the embedding of global tokens.

Roughly synchronous to BigBird, Beltagy *et al.* [78] developed the similar *Longformer* model, which exploits the lower computational complexity of the sparse attention operation to drastically increase the processable input sequence length. It uses the same three sparse attention patterns as BigBird, with the exception that dilated windows are used to further increase the receptive field of the model with increasing model depth at no additional computational cost.

Decoding-enhanced BERT with disentangled attention (DeBERTa). DeBERTa [79] refines the token embedding strategy of BERT in two ways. First, each token is explicitly represented with n relative position embeddings to all other tokens in a sequence in addition to a content embedding. The positional and content vectors are summarized in matrices and used to calculate three attention scores of a token towards another, that is content-to-content, content-to-position, and position-to-content. Content- and position-specific query and key projection matrices are used respectively. The above strategy captures crucial linguistic information on the relatedness and importance of tokens in a sequence. Second, DeBERTa incorporates absolute positional information at the very end of the architecture instead of adding it to the token representations in the beginning as done with BERT. Hence, the model is more accurately tuned to capture information on content and relative positions as only a small amount of parameters attends to the absolute positioning of tokens.

Language Understanding with Knowledge-based Embeddings (LUKE). LUKE [80] extends the masked LM

of BERT with a novel training task that accounts for entity-level tokens. This is achieved by explicitly masking entities in addition to words and giving them a distinct token embedding. Furthermore, the attention mechanism is adapted to explicitly model the relations of words to entities and vice versa. LUKE also accounts for architectural optimization since BERT by using the improved RoBERTa as its baseline.

Switch Transformer. The Switch Transformer [81] represents a sparsely-activated model that maximizes parameter count efficiently to increase model performance, while simultaneously reducing training time. More accurately, the model consists of a large number of expert models with their own parameters. In each layer the single best expert is determined for each token in the input by a routing mechanism and the resulting representations are linearly combined and passed on to the next layer. Since the size of all weight matrices is determined beforehand but the decisions of the routing mechanism are dynamic, an auxiliary loss is added to the model to encourage the router to evenly distribute the layer inputs across all available experts. Otherwise, too many tokens may be assigned to an expert, which causes those tokens to be dropped, that is not be used in the current layer.

Knowledge Embedding and Pre-trained LanguagE Representation (KEPLER). KEPLER [82] captures factual as well as linguistic knowledge in its embeddings by training on knowledge graphs and arbitrary text respectively. Concretely, the training of KEPLER is jointly governed by two loss functions expressed in a knowledge embedding task with negative sampling and the masked language modeling task of BERT. The knowledge embedding task uses entity descriptions and fixed relation embeddings extracted from knowledge graphs and subsequently maximizes the similarity of a given entity description with any neighboring entity description, while minimizing the similarity of not connected entities. At that, the projection of the entity descriptions is done with the same encoder that is used during masked language modeling. Hence, KEPLER can build on top of existing Transformer architectures. Precisely, KEPLER initializes with a RoBERTa checkpoint.

Character Architecture with No tokenization In Neural Encoders (CANINE). Clark *et al.* [83] identify explicit tokenization as a rudiment of the beginnings of artificial-neural-network-based TR. They present CANINE, an encoder-only Transformer that instead learns tokenization by operating directly on byte strings. In particular, Unicode character codepoints are converted into numerical representations by concatenating the results of multiple hashing functions. These representations are then fed to a block-wise local attention layer in order for it to learn a composition function to greater linguistic units, for example subwords. Next, strided convolution is applied to the representation vectors as a downsampling method to compensate for the greater sequence length of Unicode character codepoints in comparison to textual units, for example characters. The resulting representations are then fed through a deep stack of Transformer encoders. For downstream tasks that require the prediction of text sequences, an upsampling method is provided. The downsampled deep representations of the model are padded to the input sequence length by replicating them n times, n being the downsampling rate, and concatenating them with the representations of the local attention layer.

B. GENEALOGY

In order to trace the history of the presented TR methods we place them in a genealogy that comprises temporal information along one axis and identifies different evolutionary branches on the other axis. The latter describes four paradigms that can be traced throughout the history of TR: *size*, *context*, *efficiency*, and *multi-tasking*. Hence, any TR method can be assigned to at least one of these evolutionary branches.

Size. TR methods that fall into the *size* branch value more data and larger models over highly curated corpora and custom-built training tasks. These models consistently achieve state-of-the-art results on NLP tasks but require extensive computational resources and distributed architectures, making them inaccessible for most researchers.

Context. The representatives of the *context* branch aim at increasing the distance of textual dependencies that can be modeled. This is achieved either by allowing for longer text sequences to be processed by a model or by adapting the memory mechanism of a model, for example storing more network activations. Context information is especially valuable if either the input or the output of a model is expected to be long in a given NLP task, for example for summarization.

Efficiency. The *efficiency* branch tries to achieve high performance on a small scale. It has gained traction due to the trend of continuously creating larger, more potent models, which, however, remain inaccessible for most researchers due to immense resource costs. Common approaches are the optimization of the operations of a TR method or the transferal of the abilities of larger models to versions with a smaller footprint.

Multi-tasking. Finally, the *multi-tasking* branch aims at explicitly capturing many facets of language by simultaneously optimizing a model on various carefully crafted training tasks. Hence, finding an effective combination of training tasks is the principal objective of this branch. For instance, it could include semantic, syntactic, and orthographic tasks.

Figure 2 provides a schematic overview of the relations of the different TR methods.

An analysis of the temporal axis of the genealogy indicates that the evolution of TR can be split into two phases. The *first phase* ranges from 2013 to 2017 and the *second phase* ranges from 2019 until today. 2018 can be seen as the transition between the two phases. At the beginning of each phase stands a novel approach that changed the landscape of TR and inspired the development of manifold TR methods. This is reflected in the high number of outgoing connections of such a TR method in the genealogy. Consequently, the first phase was initiated by the word2vec models and the second phase by BERT and, in part, GPT. During the first phase, new TR

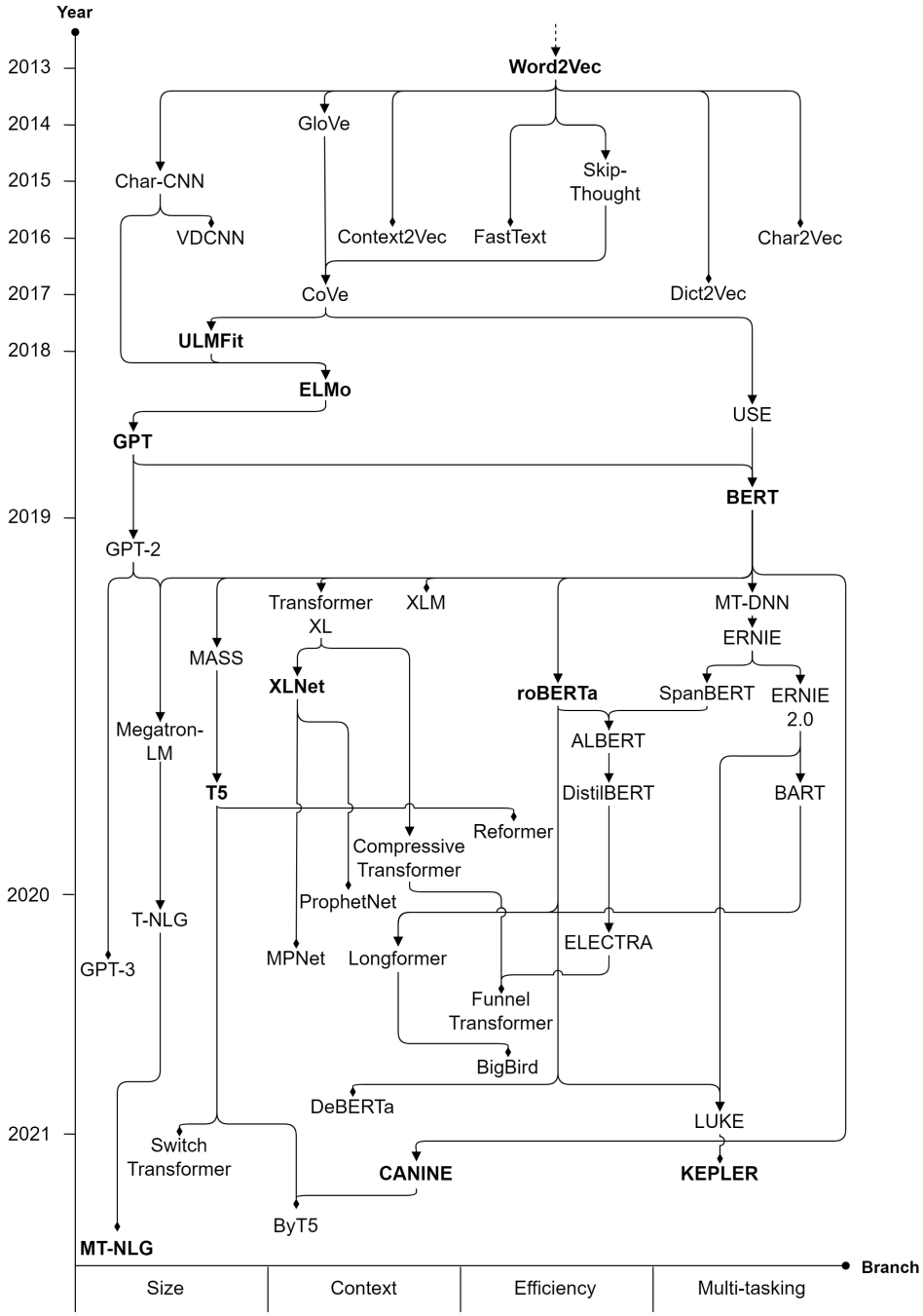


FIGURE 2. Genealogy of recent TR methods.

methods were inspired by word2vec models, yet they scarcely built on top of each other and rather branched out separately to address the four paradigms size, context, efficiency, and multi-tasking. The high amount of leaf nodes in the genealogy illustrates that the first phase was characterized by radical changes concerning model architecture and other conceptional choices. The second phase stands in stark contrast. While the TR models still branched out, the tendency was to refine and extend previous methods, in particular BERT and GPT. This emphasizes that a robust architecture was found with the Transformer [36]. An analysis of the transition period reveals two mostly disjoint *streams* that clearly show the adoption of LM in favor of previous contextualization approaches. Moreover, a split can be identified between TR methods that employ autoregressive approaches (left) and those that use autoencoder approaches (right). Taking another look at the second period, however, the two split streams are in the process of reuniting due to an increasing number of new

sequence-to-sequence architectures, which combine aspects of autoencoding and autoregression.

Finally, an illustration of the central problems in the history of TR is helpful to comprehend the motivation that drove the evolution of TR methods. In our genealogy, we highlight methods in bold that constitute the solution to such a problem and subsequently advanced the performance of NLP in general and TR in particular. It is worthwhile to mention that the transition period played a major role in shaping the evolution of TR as indicated by several consecutive bold methods. The first hurdle of modern TR was a severe computational bottleneck. Pioneering FFNN could not efficiently model the highly variant feature space of natural language and had to either reduce computation time or training data, both leading to diminishing TR capabilities. The hurdle was taken with the **word2vec** models that provided a parameter efficient architecture to make the calculation of distributional embeddings feasible for large corpora. However, the embeddings were stored in static lookup-tables. In effect, homonyms were united in a single embedding and thus made indistinguishable. The setup thus inherently discarded polysemy. In addition, the resulting embedding inevitably modeled a bias that was likely extreme and intransparent. **ELMo** received a lot of attention for solving the problem of polysemy. The approach created token embeddings on a per sample basis as a function of the entire input sequence. This allowed for the distinction of homonyms according to context information. Instead of inputting static lookup-tables, ELMo provided a dynamic embedding model that could be prepended to a task-specific architecture. However, a fundamental design principle for the training of artificial neural networks was still violated. The models could not directly map raw data to an inference, which gave space to errors and diminished performance. **ULMFiT** proposed an end-to-end approach with great success and restructured the course of the NLP domain in its wake. Recent years document that the Transformer has established itself as the best performing end-to-end architecture for NLP. With that the central problems in the NLP domain have shifted from changing the layout and combination of different model architectures to exploiting the capacities of the now ubiquitous Transformer. At the beginning of that process stand **BERT** and **GPT** which achieved high language understanding and generation capabilities respectively. Subsequently, the creators of **XLNet** aspired to combine the generative power of GPT and discriminative power of BERT into one model. **RoBERTa** focused on refining BERT to train more efficiently and effectively and managed to lay a strong foundation that has been used for the creation of new models up to more than a year later. **KEPLER** focused on explicitly integrating common sense and world knowledge to draw NLP methods near to the pragmatic layer. Meanwhile, **T5** was designed to be task-agnostic, that is applicable to any NLP problem, as a one-for-all solution. **CANINE** extended the end-to-end principle to the model vocabulary by operating on byte strings instead of predefined token vocabularies. Lastly, **MT-NLG** pushed computational boundaries with an immense amount of model parameters driven by the goal of creating an artificial intelligence that captures and thus understands all aspects of language.

V. DISCUSSION OF TAXONOMY OF TEXT REPRESENTATION METHODS

The compilation and analysis of TR methods conducted above enabled us to create a taxonomy of TR methods. It comprises the five dimensions *architecture*, *vocabulary*, *representation*, *domain-dependency*, and *training strategy* along which a classification of a given TR method can be made (see Table 3). In the following, we describe its dimensions and discuss the changes through time in each dimension to point out the current state-of-the-art. Note that the dimensions are sometimes fuzzy, for example ELMo combines two architectures, the CNN and the LSTM. A classification of all TR methods can be found in Table 4.

Architecture. The first dimension is the architecture of TR methods. The evolution in this dimension is mainly driven by the increase in computational capabilities and artificial neural networks. Early TR methods relied on *FFNN* or therein inspired simple statistical calculations. The main limitation of these approaches were their inability to model sequence information efficiently. Due to their lack of expressiveness, these approaches have been replaced by deeper and more complex models, more precisely *CNN* and *RNN*. CNN stood out with their efficiency and ability to extract high-quality features from elementary parts of text, especially characters, while RNN capitalized on the sequential nature of text. LSTM have had exceptional success by combining effective sequence modeling with stable network training. However, a step-change in NLP has been taking place with the introduction of the *Transformer* architecture. It exploits present-day computational possibilities through its parallelizable structure, and outclasses other model architectures in terms of capturing long-range dependencies in text.

Vocabulary. The vocabulary dimension describes the tokenization granularity of a TR method. The body of literature distinguishes four granularities: *byte*-level, *character*-level, *subword*-level, and *word*-level tokens. On account of a straightforward implementation, the first approaches for TR used word-level tokens. However, the low granularity led to extensive vocabularies, resulting in generalization issues. A particular problem were OOV and rare words. Character- and subword-level tokens subsequently mitigated the problem of large vocabularies and OOV words but faced the challenge of learning meaningful compositions of tokens to larger linguistic units. Nonetheless, in terms of downstream task performance they surpassed word-level methods, implying the use of word morphology to better generalize language. More specifically, the morphological information could be accessed either implicitly through the combination of character-level features or explicitly through the incorporation of subword-level tokens into the vocabulary. However, character-level approaches struggled with underfitting natural language so that subword vocabularies have become the de

TABLE 3. Taxonomy of TR methods highlighting the state-of-the-art.

Dimension	Expression			
Architecture	FFNN	CNN	RNN	Transformer
Vocabulary	Byte	Character	Subword	Word
Representation	Byte	Character	Subword	Word	Sequence
Domain-dependency	Supervised	Semi-supervised	Unsupervised	
Training Strategy	Context Compression	Autoregressive LM	Autoencoder LM	seq2seq

facto standard. Importantly, this status quo might be about to change due to highly flexible byte-level approaches. They do away with explicit vocabulary creation and, in consequence, the need for text pre-processing, allowing models to operate on any language and to learn their own token boundaries in an end-to-end fashion at the cost of computational overhead.

Representation. The representation dimension determines the units of text for which distributional representations are created by a TR method. A priori, the unit of representation of a given method cannot be of a finer granularity than its vocabulary. Contrarily, low-level token representations can be composed to represent larger linguistic units. Furthermore, the smallest linguistic unit distributional representations seem sensible for are subwords, given that morphemes are the smallest meaning bearing elements of language [84]. Nonetheless, some approaches still choose a lower representation level. Overall, TR methods represent text on a *byte*, *character*, *subword*, *word*, and *sequence* level. As with the vocabulary, TR started at the word-level because of the simple implementation. Afterwards, sequence representations were explored and performed better than word-level representations on NLP tasks that required such high-level representations, for example text classification. However, due to their aforementioned compositional flexibility, subword representations have become the de facto standard. Even so, most state-of-the-art TR methods additionally include sequence representations, because a simple composition of subwords to such a large textual unit would likely carry with it a degradation of the representation quality. That is, the meaning of a sequence can be larger than the combination of the meaning of the individual subwords [4]. This might further explain why character-level representations played only a marginal role in the evolution of TR methods. Note that while byte-level representation faces a similar problem, it benefits from the aforementioned advantages, that is a vocabulary-free and thus more integrated end-to-end training. That being said, it remains to be seen whether byte-level can improve upon subword-level representations.

Domain-dependency. The fourth dimension of TR methods is the domain-dependency. TR can either be trained in a *supervised*, *semi-supervised*, or *unsupervised* fashion. The first TR methods were unsupervised, which allowed the models to be trained on vast amounts of data. The resulting text representations embedded general aspects of language and could be used as the basis for task-specific architectures. Contrary to this development, supervised approaches were recognized to improve performance on NLP tasks by fitting text representations to the task domain. Yet, the evident drawback of supervised training was the limited in-domain data, and the approach was discarded as unsupervised corpora grew to effectively contain the entire textual internet. The necessity of training task-specific architectures from scratch when using unsupervised approaches on the one hand and the lack of training data in a supervised setting on the other hand, led to the adoption of semi-supervised TR methods. Semi-supervision combines the advantages of both previous approaches and hence established as the state-of-the-art in this dimension. A model is first trained on a huge corpus to capture the general aspects of language. Subsequently, the same model is fine-tuned on task-specific data.

Training strategy. The last dimension is the training strategy. It refers to how the distributional hypothesis is implemented to create distributional representations. Four fundamental training strategies can be distinguished: *context compression*, *autoregressive LM*, *autoencoder LM*, and *Seq2Seq*. The first approaches were FFNN-based strategies that aligned the vector representations of tokens that occurred together in fixed size windows around a position. These approaches were efficient but did not account for order information. Moreover, context-windows were typically smaller than the input sequence, limiting the context information available to these models. Although later approaches mitigated both drawbacks by employing RNN and extending the context-window to the entire input sequence respectively, order information was still not used effectively. Specifically, merely compressing order and context information into embeddings according to a sequence was not sufficient. Subsequently, autoregressive LM improved model performance by leveraging order information in explicitly learning the distribution of sequences in a corpus. However, these models introduced uni-directionality as a new limitation to the modeling of context information because only the preceding tokens to a target token were considered at any time-step. Many TR methods dealt with this problem by concatenating two opposite autoregressive LMs with good results, but autoencoder LMs made it possible to directly include bi-directional context information of an entire sequence for every position in a deep manner. Thereby, great strides were made in terms of downstream task performance on many NLP tasks, the caveat being inferior scalability and generative capabilities in comparison to autoregressive LMs. Hence, efforts were made to integrate bi-directional context into autoregressive LMs. Especially a hybrid learning strategy known as Seq2Seq has gained traction. It combines aspects of autoencoding

TABLE 4. Classification of the identified TR methods according to our taxonomy.

	Architecture				Vocabulary				Representation					Domain-dependency			Training strategy			
	FFNN	CNN	RNN	Transformer	Byte	Character	Subword	Word	Byte	Character	Subword	Word	Sequence	Supervised	Semi-supervised	Unsupervised	Context Compression	Autoregressive LM	Autoencoder LM	seq2seq
word2vec	X							X				X				X	X			
GloVe								X				X				X	X			
skip-thought	X							X					X			X	X			
char-CNN		X				X							X	X			X			
FastText	X						X				X					X	X			
char2vec			X			X						X				X	X			
context2vec			X					X				X				X	X			
VDCNN		X				X							X	X			X			
dict2vec	X							X				X				X	X			
CoVe			X				X				X		X			X				
ULMFiT			X					X				X			X			X		
USE	X			X				X					X	X		X	X			
ELMo		X	X			X						X				X		X		
GPT(1-3)				X			X				X				X			X		
BERT				X			X				X		X		X				X	
MT-DNN				X			X				X		X	X	X				X	
XLM				X			X				X		X	X	X			X	X	
TransformerXL				X			X				X				X			X		
ERNIE(1-2)				X			X				X		X		X				X	
MASS				X			X				X				X					X
XLNet				X			X				X		X		X			X		
RoBERTa				X			X				X		X		X				X	
SpanBERT				X			X				X		X		X				X	
ALBERT				X			X				X		X		X				X	
Megatron-LM				X			X				X		X		X			X	X	
DistilBERT				X			X				X		X		X				X	
T5				X			X				X			X	X					X
BART				X			X				X		X		X					X
Reformer				X			X				X				X					X
Compressive Transformer				X			X				X				X			X		
ProphetNet				X			X				X		X		X			X		
T-NLG				X			X				X				X			X		
ELECTRA				X			X				X		X		X				X	
Longformer				X		X	X			X	X		X		X			X	X	X
MPNet				X			X				X		X		X			X	X	
Funnel Transformer				X			X				X		X		X				X	X
BigBird				X			X				X		X		X				X	
DeBERTa				X			X				X		X		X				X	
LUKE				X			X				X		X		X				X	
Switch Transformer				X			X				X				X				X	
KEPLER				X			X				X		X		X				X	
CANINE				X	X		X		X				X		X			X	X	
ByT5				X	X				X					X	X					X
MT-NLG				X			X				X				X			X		

and autoregression for a flexible approach in terms of input manipulations, scalability, and downstream task applicability with competitive performance.

VI. RELATED WORK

In an attempt to survey TR, several authors have contributed to the field but did not provide a comprehensive compilation, composition, and systematization.

Cambria and White [7] use the concept of *jumping curves* to illustrate the change of NLP from lexical to compositional semantics, that is not analyzing the words but the concepts of a text. However, they do not cover specific TR methods.

Otter *et al.* [6] break down NLP into natural language modeling, morphology, and semantics. The advances in each field are discussed in the context of deep learning and are framed with real world applications. Thus, the authors place

a strong focus on task-specific models as well as architectures and digress from presenting the underlying TR methods. Furthermore, due to the publishing date of the paper, current advances are not covered.

Ferrone and Zanzotto [5] review the different modalities of TR, for example symbolic and distributed representation. They show a trade-off between expressiveness and interpretability. In particular, they examine the ability of distributed representations to exploit more aspects of texts, while being harder to interpret. Nevertheless, their research is limited to early TR methods.

Li *et al.* [85] portray the evolution of NLP from shallow to deep models. They demonstrate this development with the example of the text classification task in NLP. While they mention a variety of different TR methods, the focus is rather placed on specific model architectures relating to downstream tasks.

Zhou *et al.* [1] structure the evolution of NLP along the perspectives modeling, learning, and reasoning. Similarly to Li *et al.* [85], they give a representative overview of TR, only briefly covering concrete methods. Moreover, reasoning is not related to TR specifically but rather to the broader NLP domain.

Wang *et al.* [18] identify three challenges in the history of TR methods: words, the understanding of context, and representations for different languages. They continue to show in detail the key TR methods and concepts that helped to overcome these challenges. However, improvements beyond the key approaches are not accounted for. As a result, we adopt a representative perspective, although more fine-grained with respect to TR than given by Zhou *et al.* [1] or Li *et al.* [85].

Wang *et al.* [86] focus on illustrating the trend from static to dynamic TR. They extend the considerations from Ferrone and Zanzotto [5] by describing a selection of important TR methods in detail, including recent approaches. Moreover, they highlight the conceptual challenges the TR methods overcome to improve on previous approaches. Nonetheless, as in [18], fine-grained developments in the field of TR are not covered.

Peng and Han [87] and Duan *et al.* [88] elaborate on some of the most pervasive, recent TR methods. Yet, the brevity of the surveys is indicative of their incompleteness.

In contrast, Han *et al.* [89] provide an extensive review of pre-trained models for TR. More precisely, they group the models according to the underlying problem that motivated their conception. The identified groups are further used to visualize the interrelations between and motivations behind the models in a family tree. But in spite of the level of detail throughout the paper, the authors shed little light on approaches preceding GPT and BERT, do not explain how the presented models work in detail, and show only rudimentary interrelations between the models.

VII. CONCLUSION

TR has been evolving at an unprecedented rate. The result is a plethora of new and improved TR methods leading to a diffuse overall picture of the research field, which is confusing for novices in the domain of NLP and at least challenging for experts to keep track of. In this work, we made the effort to compile and systematize these TR methods and thus shed light on the recent evolution of TR.

In order to trace the genealogy of TR, many perspectives on the lineage of TR methods are possible. A comprehensive view can be achieved through their combination. In particular, we described the evolution through the development of conceptual choices, chronological relations, and design paradigms. In addition, the motivations behind pivotal TR methods supplement more detailed information. The end of the genealogy displays the current state-of-the-art. However, it cannot be defined as singular methods but rather as a combination of conceptual choices. More concretely, Transformer-based models that train a semi-supervised LM on subwords to form subword representations.

Looking forward it can be suspected that the pace of radical changes in the TR domain, which has been sustained over recent years, is decelerating. The Transformer architecture seems to have stopped conceptual revolutions in favor of gradual increments. Nevertheless, new approaches are constantly being proposed introducing changes that need to be evaluated.

Current developments suggest that progress will be divided into two fields. On the one hand, there is an increased interest in products based on artificial intelligence, which drives the development of task-specific architectures, models, and methods. On the other hand, resource-rich players, for example Google and Nvidia, continue to search for breakthroughs on the research side of TR. In particular, a complete understanding of natural language, that is on the lexical, syntactic, semantic and, crucially, pragmatic layer.

Unfortunately, the necessity for an enormous amount of computational resources in the latter case is closing off the research field for many. In light of this, focus may as well shift to other open issues, for example explaining distributional representations or dealing with structural bias.

REFERENCES

[1] M. Zhou, N. Duan, S. Liu, and H.-Y. Shum, "Progress in neural NLP: Modeling, learning, and reasoning," *Engineering*, vol. 6, no. 3, pp. 275–290, Mar. 2020.

[2] M. Christiansen and S. Kirby, "Language evolution: The hardest problem in science?" in *Language Evolution*. Oxford, U.K.: Oxford Univ. Press, Jul. 2003, pp. 1–15.

[3] W.-W. Guo, L.-L. Huang, and J.-S. Pan, "A review of the development and application of natural language processing," in *Genetic and Evolutionary Computing* (Advances in Intelligent Systems and Computing), J.-S. Pan, J. C.-W. Lin, Y. Liang, and S.-C. Chu, Eds. Singapore: Springer, 2020, pp. 437–443.

[4] Y. Goldberg and G. Hirst, *Neural Network Methods in Natural Language Processing*. San Rafael, CA, USA: Morgan & Claypool, 2017.

[5] L. Ferrone and F. M. Zanzotto, "Symbolic, distributed, and distributional representations for natural language processing in the era of deep learning: A survey," *Frontiers Robot. AI*, vol. 6, p. 153, Jan. 2020.

[6] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 604–624, Feb. 2020.

[7] E. Cambria and B. White, "Jumping NLP curves: A review of natural language processing research [review article]," *IEEE Comput. Intell. Mag.*, vol. 9, no. 2, pp. 48–57, May 2014.
[8] K. Mote, "Natural language processing—A survey," 2012, *arXiv:1209.6238*.
[9] R. V. Yampolskiy, "Turing test as a defining feature of AI-completeness," in *Artificial Intelligence, Evolutionary Computing and Metaheuristics: In the Footsteps of Alan Turing* (Studies in Computational Intelligence), X.-S. Yang, Ed. Berlin, Germany: Springer, 2013, pp. 3–17, doi: 10.1007/978-3-642-29694-9_1.
[10] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999.
[11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 24, 2019.
[12] J. Camacho-Collados and M. T. Pilehvar, "From word to sense embeddings: A survey on vector representations of meaning," *J. Artif. Intell. Res.*, vol. 63, pp. 743–788, Dec. 2018.
[13] A. Gudivada, D. L. Rao, and V. N. Gudivada, "Linguistics: Core concepts and principles," in *Handbook of Statistics* (Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications), vol. 38, V. N. Gudivada and C. R. Rao, Eds. Amsterdam, The Netherlands: Elsevier, Jan. 2018, pp. 3–14. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169716118300208
[14] M. Z. Kurdi, *Natural Language Processing and Computational Linguistics: Speech, Morphology and Syntax*. Hoboken, NJ, USA: Wiley, Aug. 2016.
[15] I. A. Sag, T. Baldwin, F. Bond, A. Copestake, and D. Flickinger, "Multiword expressions: A pain in the neck for NLP," in *Computational Linguistics and Intelligent Text Processing* (Lecture Notes in Computer Science), vol. 2276, G. Goos, J. Hartmanis, J. van Leeuwen, and A. Gelbukh, Eds. Berlin, Germany: Springer, 2002, pp. 1–15. [Online]. Available: http://link.springer.com/10.1007/3-540-45715-1_1
[16] K. Korta and J. Perry, "Pragmatics," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed. Stanford, CA, USA: Stanford Univ., 2020. [Online]. Available: https://plato.stanford.edu/archives/spr2020/entries/pragmatics/
[17] D. Yin, L. Dong, H. Cheng, X. Liu, K.-W. Chang, F. Wei, and J. Gao, "A survey of knowledge-intensive NLP with pre-trained language models," 2022, *arXiv:2202.08772*.
[18] S. Wang, W. Zhou, and C. Jiang, "A survey of word embeddings based on deep learning," *Computing*, vol. 102, no. 3, pp. 717–740, Mar. 2020, doi: 10.1007/s00607-019-00768-7.
[19] P. M. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, p. 78–87, Oct. 2012, doi: 10.1145/2347736.2347755.
[20] Z. S. Harris, "Distributional structure," *Word*, vol. 10, nos. 2–3, pp. 146–162, Aug. 1954, doi: 10.1080/00437956.1954.11659520.
[21] J. R. Firth, "A synopsis of linguistic theory 1930–1955," *Stud. Linguistic Anal.*, vols. 1952–1956, pp. 1–32, Aug. 1957.
[22] S. Liu, P.-T. Bremer, J. J. Thiagarajan, V. Srikumar, B. Wang, Y. Livnat, and V. Pascucci, "Visual exploration of semantic relationships in neural word embeddings," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 1, pp. 553–562, Jan. 2018.
[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.
[24] Z. Liu, Y. Lin, and M. Sun, "Word representation," in *Representation Learning for Natural Language Processing*, Z. Liu, Y. Lin, and M. Sun, Eds. Singapore: Springer, 2020, pp. 13–41, doi: 10.1007/978-981-15-5573-2_2.
[25] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electron. Markets*, vol. 31, no. 3, pp. 685–695, Sep. 2021, doi: 10.1007/s12525-021-00475-2.
[26] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing: A survey," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 3, pp. 1–41, Jun. 2020, doi: 10.1145/3374217.
[27] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.
[28] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," 2016, *arXiv:1606.01781*.
[29] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018, *arXiv:1802.05365*.
[30] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.
[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, Nov. 2016.
[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.
[33] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014, *arXiv:1406.1078*.
[34] A. Galassi, M. Lippi, and P. Torroni, "Attention in natural language processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4291–4308, Oct. 2021, doi: 10.1109/TNNLS.2020.3019893.
[35] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. U. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
[37] D. Cer, Y. Yang, S.-Y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope, and R. Kurzweil, "Universal sentence encoder for English," in *Proc. Conf. Empirical Methods Natural Lang. Process., Syst. Demonstrations*, 2018, pp. 169–174. [Online]. Available: https://aclanthology.org/D18-2029
[38] S. K. Boell and D. Cecez-Kecmanovic, "A hermeneutic approach for conducting literature reviews and literature searches," *Commun. Assoc. Inf. Syst.*, vol. 34, no. 1, pp. 257–286, 2014. [Online]. Available: https://aisel.aisnet.org/cais/vol34/iss1/12
[39] H. M. Cooper, "Organizing knowledge syntheses: A taxonomy of literature reviews," *Knowl. Soc.*, vol. 1, no. 1, p. 104, Mar. 1988, doi: 10.1007/BF03177550.
[40] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 2. Red Hook, NY, USA: Curran Associates, 2013, pp. 3111–3119.
[41] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1532–1543. [Online]. Available: http://aclweb.org/anthology/D14-1162
[42] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in Neural Information Processing Systems*, vol. 28, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/f442d33fa06832082290ad8544a8da27-Paper.pdf
[43] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, vol. 28, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf
[44] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," 2016, *arXiv:1607.04606*.
[45] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "FastText. Zip: Compressing text classification models," 2016, *arXiv:1612.03651*.
[46] K. Cao and M. Rei, "A joint model for word embedding and word morphology," in *Proc. 1st Workshop Represent. Learn. NLP*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 18–26. [Online]. Available: https://aclanthology.org/W16-1603
[47] O. Melamud, J. Goldberger, and I. Dagan, "Context2Vec: Learning generic context embedding with bidirectional LSTM," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.* Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 51–61. [Online]. Available: https://www.aclweb.org/anthology/K16-1006

[48] J. Tissier, C. Gravier, and A. Habrard, "Dict2vec: Learning word embeddings using lexical dictionaries," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 254–263. [Online]. Available: https://www.aclweb.org/anthology/D17-1024

[49] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Red Hook, NY, USA: Curran Associates, 2017, pp. 6297–6308.

[50] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," 2018, *arXiv:1801.06146*.

[51] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI, San Francisco, CA, USA, Tech. Rep., 2018, p. 12. [Online]. Available: https://paperswithcode.com/paper/improving-language-understanding-by

[52] T. B. Brown *et al.*, "Language models are few-shot learners," Jul. 2020, *arXiv:2005.14165*.

[53] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," May 2019, *arXiv:1901.11504*.

[54] A. Conneau and G. Lample, "Cross-lingual language model pretraining," in *Advances in Neural Information Processing Systems*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/c04c19c2c2474dbf5f7ac4372c5b9af1-Paper.pdf

[55] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," Jun. 2019, *arXiv:1901.02860*.

[56] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap, "Compressive transformers for long-range sequence modelling," Nov. 2019, *arXiv:1911.05507*.

[57] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu, "ERNIE: Enhanced representation through knowledge integration," Apr. 2019, *arXiv:1904.09223*.

[58] Y. Sun, S. Wang, Y. Li, S. Feng, H. Tian, H. Wu, and H. Wang, "ERNIE 2.0: A continual pre-training framework for language understanding," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 5, pp. 8968–8975. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6428

[59] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "MASS: Masked sequence to sequence pre-training for language generation," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, K. Chaudhuri and R. Salakhutdinov, Eds., Jun. 2019, pp. 5926–5936. [Online]. Available: https://proceedings.mlr.press/v97/song19d.html

[60] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Advances in Neural Information Processin Systems*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html

[61] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," Jul. 2019, *arXiv:1907.11692*.

[62] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "SpanBERT: Improving pre-training by representing and predicting spans," *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 64–77, Oct. 2020, doi: 10.1162/tacl_a_00300.

[63] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," Feb. 2019, *arXiv:1909.11942*.

[64] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "ZeRO: Memory optimizations toward training trillion parameter models," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal. (SC)*. Piscataway, NJ, USA: IEEE Press, Nov. 2020, pp. 1–16.

[65] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training multi-billion parameter language models using model parallelism," 2019, *arXiv:1909.08053*.

[66] C. Rosset. (Feb. 2020). *Turing-NLG: A 17-Billion-Parameter Language Model by Microsoft*. [Online]. Available: https://www.microsoft.com/enus/research/blog/turing-nlg-a-17-billion-parameterlanguage-model-by-microsoft/

[67] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, E. Zhang, R. Child, R. Y. Aminabadi, J. Bernauer, X. Song, M. Shoeybi, Y. He, M. Houston, S. Tiwary, and B. Catanzaro, "Using DeepSpeed and megatron to train megatron-turing NLG 530B, a large-scale generative language model," Feb. 2022, *arXiv:2201.11990*.

[68] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," Feb. 2020, *arXiv:1910.01108*.

[69] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[70] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, and C. Raffel, "ByT5: Towards a token-free future with pre-trained byte-to-byte models," *CoRR*, vol. abs/2105.13626, pp. 1–16, May 2021.

[71] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019, *arXiv:1910.13461*.

[72] N. Kitaev, Å. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," Oct. 2020, *arXiv:2001.04451*.

[73] W. Qi, Y. Yan, Y. Gong, D. Liu, N. Duan, J. Chen, R. Zhang, and M. Zhou, "ProphetNet: Predicting future N-gram for sequence-to-sequencepre-training," in *Proc. Findings Assoc. Comput. Linguistics EMNLP*. Stroudsburg, PA, USA: Association for Computational Linguistics, Nov. 2020, pp. 2401–2410. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.217

[74] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training text encoders as discriminators rather than generators," Mar. 2020, *arXiv:2003.10555*.

[75] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "MPNet: Masked and permuted pre-training for language understanding," in *Advances in Neural Information Processing Systems*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds. Red Hook, NY, USA: Curran Associates, 2020, pp. 16857–16867. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/c3a690be93aa602ee2dc0ccab5b7b67e-Paper.pdf

[76] Z. Dai, G. Lai, Y. Yang, and Q. Le, "Funnel-transformer: Filtering out sequential redundancy for efficient language processing," in *Advances in Neural Information Processing Systems*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds. Red Hook, NY, USA: Curran Associates, 2020, pp. 4271–4282. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/2cd2915e69546904e4e5d4a2ac9e1652-Abstract.html

[77] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed, "Big Bird: Transformers for longer sequences," in *Advances in Neural Information Processing Systems*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds. Red Hook, NY, USA: Curran Associates, 2020, pp. 17283–17297. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html

[78] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," 2020, *arXiv:2004.05150*.

[79] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," in *Proc. Int. Conf. Learn. Represent.*, May 2021, pp. 1–23. [Online]. Available: https://www.microsoft.com/enus/research/publication/deberta-decoding-enhancedbert-with-disentangled-attention-2/

[80] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto, "LUKE: Deep contextualized entity representations with entity-aware self-attention," Oct. 2020, *arXiv:2010.01057*.

[81] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Deep Learn. Rev.*, vol. 23, no. 120, pp. 1–39, Jan. 2021.

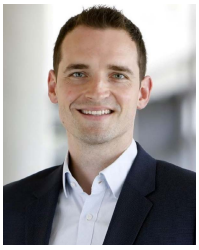
[82] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li, and J. Tang, "KEPLER: A unified model for knowledge embedding and pre-trained language representation," *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 176–194, Feb. 2021, doi: 10.1162/tacl_a_00360.

[83] J. H. Clark, D. Garrette, I. Turc, and J. Wieting, "Canine: Pre-training an efficient tokenization-free encoder for language representation," *Trans. Assoc. Comput. Linguistics*, vol. 10, pp. 73–91, Jan. 2022, doi: 10.1162/tacl_a_00448.

[84] D. Jurafsky and J. H. Martin, *Speech Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st ed. Hoboken, NJ, USA: Prentice-Hall, 2000.
[85] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. S. Yu, and L. He, "A survey on text classification: From traditional to deep learning," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 2, pp. 1–41, Apr. 2022, doi: 10.1145/3495162.
[86] Y. Wang, Y. Hou, W. Che, and T. Liu, "From static to dynamic word representations: A survey," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 7, pp. 1611–1630, Jul. 2020, doi: 10.1007/s13042-020-01069-8.
[87] J. Peng and K. Han, "Survey of pre-trained models for natural language processing," in *Proc. Int. Conf. Electron. Commun., Internet Things Big Data (ICEIB)*, Dec. 2021, pp. 277–280.
[88] J. Duan, H. Zhao, Q. Zhou, M. Qiu, and M. Liu, "A study of pre-trained language models in natural language processing," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Nov. 2020, pp. 116–121.
[89] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang, and W. Han, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, Jan. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000231

PHILIPP SIEBERS completed his studies of business informatics at the Technische Universität Dresden, where he specialized in the field of data science. Having provided solutions for several computer vision projects with industry stakeholders, most recently, he focused on the field of natural language processing, which not only provided the framework for his thesis, but is currently being applied to power a start-up platform aimed at continued learning.

CHRISTIAN JANIESCH is currently a Full Professor of enterprise computing at TU Dortmund University. Before, he worked full-time at various institutions, including the Westfälische Wilhelms-Universität in Münster, the SAP Research Center Brisbane of SAP Australia Pty Ltd., the Karlsruhe Institute of Technology, and the Julius-Maximilians-Universität Würzburg. His work has appeared in journals, such as the *Journal of the Association for Information Systems*, *Communications of the Association for Information Systems*, *International Journal of Information Management*, *Information & Management*, *Business & Information Systems Engineering*, *Information Systems*, *Decision Support Systems*, and *Future Generation Computer Systems*, as well as in various major international conferences, including ICIS, ECIS, BPM, WI, and HICSS, and has been registered as U.S. patents. He has authored over 150 scholarly publications. His research interests include intelligent systems at the intersection of business process management and artificial intelligence with frequent applications in the Industrial Internet of Things. He is on the BPM Department Editorial Board for *BISE* journal, as well as the Editorial Boards of *IJMR* and *JBA*.



PATRICK ZSCHECH received the Ph.D. degree from the Technische Universität Dresden. He is currently an Assistant Professor of intelligent information systems at Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany. Before that, he worked with the IT service provider Robotron Datenbank-Software GmbH as a Project Member and an Instructor for data science qualification programs. The focus of his research is on business analytics, machine learning, and artificial intelligence with a particular interest in the design, analysis, and use of intelligent information systems. His results have been published in leading IS journals, such as *Decision Support Systems*, *Business & Information Systems Engineering*, *Electronic Markets*, and *Journal of Business Analytics*, and have been presented at international conferences, such as ICIS, ECIS, and WI.