# The Unstoppable Rise of Computational Linguistics in Deep Learning

**James Henderson**
Idiap Research Institute, Switzerland
james.henderson@idiap.ch

## Abstract

In this paper, we trace the history of neural networks applied to natural language understanding tasks, and identify key contributions which the nature of language has made to the development of neural network architectures. We focus on the importance of variable binding and its instantiation in attention-based models, and argue that Transformer is not a sequence model but an induced-structure model. This perspective leads to predictions of the challenges facing research in deep learning architectures for natural language understanding.

## 1 Introduction

When neural networks first started being applied to natural language in the 1980s and 90s, they represented a radical departure from standard practice in computational linguistics. Connectionists had vector representations and learning algorithms, and they didn't see any need for anything else. Everything was a point in a vector space, and everything about the nature of language could be learned from data. On the other hand, most computational linguists had linguistic theories and the poverty-of-the-stimulus argument. Obviously some things were learned from data, but all the interesting things about the nature of language had to be innate.

A quarter century later, we can say two things with certainty: they were both wrong. Vector-space representations and machine learning algorithms are much more powerful than was thought. Much of the linguistic knowledge which computational linguists assumed needed to be innate can in fact be learned from data. But the unbounded discrete structured representations they used have not been replaced by vector-space representations. Instead, the successful uses of neural networks in computational linguistics have replaced specific pieces of computational-linguistic models with new neural network architectures which bring together continuous vector spaces with structured representations in ways which are novel for both machine learning and computational linguistics.

Thus, the great progress which we have made through the application of neural networks to natural language processing should not be viewed as a conquest, but as a compromise. As well as the unquestionable impact of machine learning research on NLP, the nature of language has had a profound impact on progress in machine learning. In this paper we trace this impact, and speculate on future progress and its limits.

We start with a sketch of the insights from grammar formalisms about the nature of language, with their multiple levels, structured representations and rules. The rules were soon learned with statistical methods, followed by the use of neural networks to replace symbols with induced vectors, but the most effective models still kept structured representations, such as syntactic trees. More recently, attention-based models have replaced hand-coded structures with induced structures. The resulting models represent language with multiple levels of structured representations, much as has always been done. Given this perspective, we identify remaining challenges in learning language from data, and its possible limitations.

## 2 Grammar Formalisms versus Connectionism

### 2.1 Grammar Formalisms

Our modern understanding of the computational properties of language started with the introduction of grammar formalisms. Context Free Grammars (Chomsky, 1959) illustrated how a formal system could model the infinite generative capacity of language with a bounded grammar. This formalism soon proved inadequate to account for the diversity



*Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6294–6306
July 5 - 10, 2020. 

of phenomena in human languages, and a number of linguistically-motivated grammar formalisms were proposed (e.g HPSG (Pollard and Sag, 1987), TAG (Joshi, 1987), CCG (Steedman, 2000)).

All these grammar formalisms shared certain properties, motivated by the understanding of the nature of languages in Linguistics. They all postulate representations which decompose an utterances into a set of sub-parts, with labels of the parts and a structure of inter-dependence between them. And they all assume that this decomposition happens at multiple levels of representation. For example that spoken utterances can be decomposed into sentences, sentences can be decomposed into words, words can be decomposed into morphemes, and morphemes can be decomposed into phonemes, before we reach the observable sound signal. In the interests of uniformity, we will refer to the subparts in each level of representation as its *entities*, their labels as their *properties*, and their structure of inter-dependence as their *relations*. The structure of inter-dependence between entities at different levels will also be referred to as relations.

In addition to these representations, grammar formalisms include specifications of the allowable structures. These may take the form of hard constraints or soft objectives, or of deterministic rules or stochastic processes. In all cases, the purpose of these specifications is to account for the regularities found in natural languages. In the interests of uniformity, we will refer to all these different kinds of specifications of allowable structures as *rules*. These rules may apply within or between levels of representation.

In addition to explicit rules, computational linguistic formalisms implicitly make claims about the regularities found in natural languages through their expressive power. Certain types of rules simply cannot be specified, thus claiming that such rules are not necessary to capture the regularities found in any natural language. These claims differ across formalisms, but the study of the expressive power of grammar formalisms have identified certain key principles (Joshi et al., 1990). Firstly, that the set of rules in a given grammar is bounded. This in turn implies that the set of properties and relations in a given grammar is also bounded.

But language is unbounded[1] in nature, since sentences and texts can be arbitrarily long. Grammar formalisms capture this unboundedness by allowing an unbounded number of entities in a representation, and thus an unbounded number of rule applications. It is generally accepted that the number of entities grows linearly with the length of the sentence (Joshi et al., 1990), so each level can have at most a number of entities which is linear in the number of entities at the level(s) below.

Computational linguistic grammar formalisms also typically assume that the properties and relations are discrete, called symbolic representations. These may be atomic categories, as in CFGs, TAGs, CCG and dependency grammar, or they may be feature structures, as in HPSG.

### 2.2 Connectionism

Other researchers who were more interested in the computational properties of neurological systems found this reliance on discrete categorical representations untenable. Processing in the brain used real-valued representations distributed across many neurons. Based on successes following the development of multi-layered perceptrons (MLPs) (Rumelhart et al., 1986b), an approach to modelling cognitive phenomena was developed called connectionism. Connectionism uses vector-space representations to reflect the distributed continuous nature of representations in the brain. Similarly, their rules are specified with vectors of continuous parameters. MLPs are so powerful that they are arbitrary function approximators (Hornik et al., 1989). And thanks to backpropagation learning (Rumelhart et al., 1986a) in neural network models, such as MLPs and Simple Recurrent Networks (SRNs) (Elman, 1990), these vector-space representations and rules could be learned from data.

The ability to learn powerful vector-space representations from data led many connectionist to argue that the complex discrete structured representations of computational linguistics were neither necessary nor desirable (e.g. Smolensky (1988, 1990); Elman (1991); Miikkulainen (1993); Seidenberg (2007)). Distributed vector-space representations were thought to be so powerful that there was no need for anything else. Learning from data made linguistic theories irrelevant. (See also (Collobert and Weston, 2008; Collobert et al., 2011; Sutskever et al., 2014) for more recent incarnations.)

The idea that vector-space representations are adequate for natural language and other cognitive phenomena was questioned from several directions.

[1] A set of things (e.g. the sentences of a language) have unbounded size if for any finite size there is always some element in the set which is larger than that.

From neuroscience, researchers questioned how a simple vector could encode features of more than one thing at a time. If we see a red square together with a blue triangle, how do we represent the difference between that and a red triangle with a blue square, since the vector elements for red, blue, square and triangle would all be active at the same time? This is known as the variable binding problem, so called because variables are used to do this binding in symbolic representations, as in $red(x) \wedge triangle(x) \wedge blue(y) \wedge square(y)$. One proposal has been that the precise timing of neuron activation spikes could be used to encode variable binding, called Temporal Synchrony Variable Binding (von der Malsburg, 1981; Shastri and Ajjanagadde, 1993). Neural spike trains have both a phase and a period, so the phase could be used to encode variable binding while still allowing the period to be used for sequential computation. This work indicated how entities could be represented in a neurally-inspired computational architecture.

The adequacy of vector-space representations was also questioned based on the regularities found in natural language. In particular, Fodor and Pylyshyn (1988) argued that connectionist architectures were not adequate to account for regularities which they characterised as *systematicity* (see also (Smolensky, 1990; Fodor and McLaughlin, 1990)). In essence, systematicity requires that learned rules generalise in a way that respects structured representations. Here again the issue is representing multiple entities at the same time, but with the additional requirement of representing the structural relationships between these entities. Only rules which are parameterised in terms of such representations can generalise in a way which accounts for the generalisations found in language.

Early work on neural networks for natural language recognised the significance of variable binding for solving the issues with systematicity (Henderson, 1996, 2000). Henderson (1994, 2000) argued that extending neural networks with temporal synchrony variable binding made them powerful enough to account for the regularities found in language. Using time to encode variable bindings means that learning could generalise in a linguistically appropriate way (Henderson, 1996), since rules (neuronal synapses) learned for one variable (time) would systematically generalise to other variables. Although relations were not stored explicitly, it was claimed that for language understanding it is adequate to recover them from the features of the entities (Henderson, 1994, 2000). But these arguments were largely theoretical, and it was not clear how they could be incorporated in learning-based architectures.

### 2.3 Statistical Models

Although researchers in computational linguistics did not want to abandon their representations, they did recognise the importance of learning from data. The first successes in this direction came from learning rules with statistical methods, such as part-of-speech tagging with hidden Markov models. For syntactic parsing, the development of the Penn Treebank led to many statistical models which learned the rules of grammar (Collins, 1997, 1999; Charniak, 1997; Ratnaparkhi, 1999).

These statistical models were very successful at learning from the distributions of linguistic representations which had been annotated in the corpus they were trained on. But they still required linguistically-motivated designs to work well. In particular, feature engineering is necessary to make sure that these statistical machine-learning method can search a space of rules which is sufficiently broad to include good models but sufficiently narrow to allow learning from limited data.

## 3 Inducing Features of Entities

Early work on neural networks for natural language recognised the potential of neural networks for learning the features as well, replacing feature engineering. But empirically successful neural network models for NLP were only achieved with approaches where the neural network was used to model one component within an otherwise traditional symbolic NLP model.

The first work to achieve empirical success in comparison to non-neural statistical models was work on language modelling. Bengio et al. (2001, 2003) used an MLP to estimate the parameters of an n-gram language model, and showed improvements when interpolated with a statistical n-gram language model. A crucial innovation of this model was the introduction of word embeddings. The idea that the properties of a word could be represented by a vector reflecting the distribution of the word in text was introduced earlier in non-neural statistical models (e.g. (Deerwester et al., 1990; Schütze, 1993; Burgess, 1998; Padó and Lapata, 2007; Erk, 2010)). This work showed that similarity in the

| PTB Constituents | | | | |
|---|---|---|---|---|
| model | | LP | LR | F1 |
| Costa et al. (2001) | PoS | 57.8 | 64.9 | 61.1 |
| Henderson (2003) | PoS | 83.3 | 84.3 | 83.8 |
| Henderson (2003) | | 88.8 | 89.5 | 89.1 |
| Henderson (2004) | | 89.8 | 90.4 | 90.1 |
| Vinyals et al. (2015) seq2seq | | | | <70 |
| Vinyals et al. (2015) attn | | | | 88.3 |
| Vinyals et al. (2015) seq2seq | semisup | | | 90.5 |

| CoNLL09 Dependencies | | |
|---|---|---|
| model (transition-based) | UAS | LAS |
| Titov and Henderson (2007a)* | 91.44 | 88.65 |
| Chen and Manning (2014)* | 89.17 | 86.49 |
| Yazdani and Henderson (2015) | 90.75 | 88.14 |

| Stanford Dependencies | | |
|---|---|---|
| model (transition-based) | UAS | LAS |
| Chen and Manning (2014) | 91.80 | 89.60 |
| Dyer et al. (2015) | 93.10 | 90.90 |
| Andor et al. (2016) | 94.61 | 92.79 |
| Kiperwasser and Goldberg (2016) | 93.9 | 91.9 |
| Mohammadshahi and Henderson (2019) BERT | 95.63 | 93.81 |

Table 1: Some neural network parsing results on Penn Treebank WSJ. LP/LR/F1: labelled constituent precision/recall/F-measure. UAS/LAS: unlabelled/labelled dependency accuracy. *results reported in (Yazdani and Henderson, 2015).

resulting vector space is correlated with semantic similarity. Learning vector-space representations of words with neural networks (rather than SVD) have showed similar effects (e.g. (Turian et al., 2010; Mikolov et al., 2013; Levy et al., 2015; Pennington et al., 2014)), resulting in impressive improvements for many NLP tasks.

More recent work has used neural network language models to learn context-dependent embeddings of words. We will refer to such context-dependent embeddings as *token embeddings*. For example, Peters et al. (2018) train a stacked BiLSTM language model, and these token embeddings have proved effective in many tasks. More such models will be discussed below.

For syntactic parsing, early connectionist approaches (Jain, 1991; Miikkulainen, 1993; Ho and Chan, 1999; Costa et al., 2001) had limited success. The first neural network models to achieve empirical success used a recurrent neural network to model the derivation structure of a traditional syntactic constituency parser (Henderson, 2003, 2004). The recurrent neural network learns to model the sequence of parser actions, estimating the probability of the next parser action given the history of previous parser actions. This allows the decoding algorithm from the traditional parsing model to be used to efficiently search the space of possible parses. These models have also been applied to syntactic dependency parsing (Titov and Henderson, 2007b; Yazdani and Henderson, 2015) and joint syntactic-semantic dependency parsing (Henderson et al., 2013).

Crucially, these neural networks do not model the sequence of parser decisions as a flat sequence, but instead model the derivation structure it specifies. A derivation structure includes relationships for the inter-dependencies between nodes in the parse tree. The pattern of interconnections between hidden layers of the recurrent neural network (henceforth referred to as the *model structure*) is designed to follow locality in this derivation structure, thereby giving the neural network a linguistically appropriate inductive bias. More recently, Dyer et al. (2015) provide a more direct relationship between the derivation structure and the model structure with their StackLSTM parsing model.

In all these models, the use of recurrent neural networks allows arbitrarily large parse structures to be modelled without making any hard independence assumptions, in contrast to non-neural statistical models. Feed-forward neural networks have also been applied to modelling the derivation structure (Chen and Manning, 2014), but the accuracy is worse than using recurrent models (see Table 1), presumably because such models suffer from the need to make hard independence assumptions.

Representing the parse tree as a derivation sequence, rather than a derivation structure, makes it possible to define syntactic parsing as a sequence-to-sequence problem, mapping the sentence to its parse sequence. If a neural network architecture for modelling sequences (called *seq2seq* models) can perform well at this task, then maybe the structured linguistic representations of natural language are not necessary (contrary to Fodor and Pylyshyn (1988)), not even to predict those structures. Vinyals et al. (2015) report very poor results for seq2seq models when trained on the standard dataset, but good results when trained on very large automatically-parsed corpora (see Table 1 *semisup*). They only achieve good results with the limited standard dataset by adding attention, which we will argue below makes the model no longer a seq2seq model. This indicates that structured representations really do capture important generalisations about language.[2]

[2]See (Collobert and Weston, 2008; Collobert et al., 2011) for an earlier related line of work.

In contrast to seq2seq models, there have also been neural network models of parsing which directly represent linguistic structure, rather than just derivation structure, giving them induced vector representations which map one-to-one with the entities in the linguistic representation. Typically, a recursive neural network is used to compute embeddings of syntactic constituents bottom-up. Dyer et al. (2015) showed improvements by adding these representations to a model of the derivation structure. Socher et al. (2013a) only modelled the linguistic structure, making it difficult to do decoding efficiently. But the resulting induced constituent embeddings have a clear linguistic interpretation, making it easier to use them within other tasks, such as sentiment analysis (Socher et al., 2013b). Similarly, models based on Graph Convolutional Networks have induced embeddings with clear linguistic interpretations within pre-defined model structures (e.g. (Marcheggiani and Titov, 2017; Marcheggiani et al., 2018)).

All these results demonstrate the incredible effectiveness of inducing vector-space representations with neural networks, relieving us from the need to do feature engineering. But neural networks do not relieve us of the need to understand the nature of language when designing our models. Instead of feature engineering, these results show that the best accuracy is achieved by engineering the inductive bias of deep learning models through their model structure. By designing a hand-coded model structure which reflects the linguistic structure, locality in the model structure can reflect locality in the linguistic structure. The neural network then induces features of the entities in this model structure.

## 4 Inducing Relations between Entities

With the introduction of attention-based models, the model structure can now be learned. By choosing the nodes to be linguistically-motivated entities, learning the model structure in effect learns the statistical inter-dependencies between entities, which is what we have been referring to as relations.

### 4.1 Attention-Based Models and Variable Binding

The first proposal of an attention-based neural model learned a soft alignment between the target and source words in neural machine translation (NMT) (Bahdanau et al., 2015). The model structure of the source sentence encoder and the model structure of the target sentence decoder are both flat sequences, but when each target word is generated, it computes attention weights over all source words. These attention weights directly express how target words are correlated with source words, and in this sense can be seen as a soft version of the alignment structure. In traditional statistical machine translation, this alignment structure is determined with a separate alignment algorithm, and then frozen while training the model. In contrast, the attention-based NMT model learns the alignment structure jointly with learning the encoder and decoder, inside the deep learning architecture (Bahdanau et al., 2015).

This attention-based approach to NMT was also applied to mapping a sentence to its syntactic parse (Vinyals et al., 2015). The attention function learns the structure of the relationship between the sentence and its syntactic derivation sequence, but does not have any representation of the structure of the syntactic derivation itself. Empirical results are much better than their seq2seq model (Vinyals et al., 2015), but not as good as models which explicitly model both structures (see Table 1).

The change from the sequential LSTM decoders of previous NMT models to LSTM decoders with attention seems like a simple addition, but it fundamentally changes the kinds of generalisations which the model is able to learn. At each step in decoding, the state of a sequential LSTM model is a single vector, whereas adding attention means that the state needs to include the unboundedly large set of vectors being attended to. This use of an unbounded state is more similar to the above models with predefined model structure, where an unboundedly large stack is needed to specify the parser state. This change in representation leads to a profound change in the generalisations which can be learned. Parameterised rules which are learned when paying attention to one of these vectors (in the set or in the stack) automatically generalise to the other vectors. In other words, attention-based models have variable binding, which sequential LSTMs do not. Each vector represents the features for one entity, multiple entities can be kept in memory at the same time, and rules generalise across these entities. In this sense it is wrong to refer to attention-based models as sequence models; they are in fact *induced-structure* models. We will expand on this perspective in the rest of this section.

### 4.2 Transformer and Systematicity

The generality of attention as a structure-induction method soon became apparent, culminated in the development of the Transformer architecture (Vaswani et al., 2017). Transformer has multiple stacked layers of self-attention (attention to the other words in the same sequence), interleaved with nonlinear functions applied to individual vectors. Each attention layer has multiple attention heads, allowing each head to learn a different type of relation. A Transformer-encoder has one column of stacked vectors for each position in the input sequence, and the model parameters are shared across positions. A Transformer-decoder adds attention over an encoded text, and predicts words one at a time after encoding the prefix of previously generated words.

Although it was developed for encoding and generating sequences, in Transformer the sequential structure is not hard-coded into the model structure, unlike previous models of deep learning for sequences (e.g. LSTMs (Hochreiter and Schmidhuber, 1997) and CNNs (LeCun and Bengio, 1995)). Instead, the sequential structure is input in the form of position embeddings. In our formulation, position embeddings are just properties of individual entities (typically words or subwords). As such, these inputs facilitate learning about absolute positions. But they are also designed to allow the model to easily calculate relative position between entities. This allows the model's attention functions to learn to discover the relative position structure of the underlying sequence. In fact, explicitly inputting relative position relations as embeddings into the attention functions works even better (Shaw et al., 2018) (discussed further below). Whether input as properties or as relations, these inputs are just features, not hard-coded model structure. The attention weight functions can then learn to use these features to induce their own structure.

The appropriateness and generality for natural language of the Transformer architecture became even more apparent with the development of pretrained Transformer models like BERT (Devlin et al., 2019). BERT models are large Transformer models trained mostly on a masked language model objective, as well as a next-sentence prediction objective. After training on a very large amount of unlabelled text, the resulting pretrained model can be fine tuned for various tasks, with very impressive improvements in accuracy across a wide variety of tasks. The success of BERT has led to various analyses of what it has learned, including the structural relations learned by the attention functions. Although there is no exact mapping from these structures to the structures posited by linguistics, there are clear indications that the attention functions are learning to extract linguistic relations (Voita et al., 2019; Tenney et al., 2019; Reif et al., 2019).

With variable binding for the properties of entities and attention functions for relations between entities, Transformer can represent the kinds of structured representations argued for above. With parameters shared across entities and sensitive to these properties and relations, learned rules are parameterised in terms of these structures. Thus Transformer is a deep learning architecture with the kind of generalisation ability required to exhibit systematicity, as in (Fodor and Pylyshyn, 1988).

Interestingly, the relations are not stored explicitly. Instead they are extracted from pairs of vectors by the attention functions, as with the use of position embeddings to compute relative position relations. For the model to induce its own structure, lower levels must learn to embed its relations in pairs of token embeddings, which higher levels of attention then extract.

That Transformer learns to embed relations in pairs of token embeddings is apparent from recent work on dependency parsing (Kondratyuk and Straka, 2019; Mohammadshahi and Henderson, 2019, 2020). Earlier models of dependency parsing successfully use BiLSTMs to embed syntactic dependencies in pairs of token embeddings (e.g. (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016)), which are then extracted to predict the dependency tree. Mohammadshahi and Henderson (2019, 2020) use their proposed Graph-to-Graph Transformer to encode dependencies in pairs of token embeddings, for transition-based and graph-based dependency parsing respectively. Graph-to-Graph Transformer also inputs previously predicted dependency relations into its attention functions (like relative position encoding (Shaw et al., 2018)). These parsers achieve state of the art accuracies, indicating that Transformer finds it easy to input and predict syntactic dependency relations via pairs of token embeddings. Interestingly, initialising the model with pretrained BERT results in large improvements, indicating that BERT representations also encode syntactically-relevant

relations in pairs of token embeddings.

### 4.3 Nonparametric Representations

As we have seen, the problem with vector-space models is not simply about representations, but about the way learned rules generalise. In work on grammar formalisms, generalisation is analysed by looking at the unbounded case, since any bounded case can simply be memorised. But the use of continuous representations does not fit well with the theory of grammar formalisms, which assumes a bounded vocabulary of atomic categories. Instead we propose an analysis of the generalisation abilities of Transformer in terms of theory from machine learning, Bayesian nonparametric learning (Jordan, 2010). We argue that the representations of Transformer are the minimal nonparametric extension of a vector space.

To connect Transformer to Bayesian probabilities, we assume that a Transformer representation can be thought of as the parameters of a probability distribution. This is natural, since a model's state represents a belief about the input, and in Bayesian approaches beliefs are probability distributions. From this perspective, computing a representation is inferring the parameters of a probability distribution from the observed input. This is analogous to Bayesian learning, where we infer the parameters of a distribution over models from observed training data. In this section, we outline how theory from Bayesian learning helps us understand how the representations of Transformer lead to better generalisation.

We do not make any specific assumptions about what probability distributions are specified by a Transformer representation, but it is useful to keep in mind an example. One possibility is a mixture model, where each vector specifies the parameters of a multi-dimensional distribution, and the total distribution is the weighted sum across the vectors of these distributions. For example, we can interpret the vectors $x{=}x_1, \ldots, x_n$ in a Transformer's representation as specifying a belief about the queries $q$ that will be received from a downstream attention function, as in:

$$
\begin{aligned}
P(q|x) &= \sum_i P(i|x)\, P(q|x_i) \\
P(i|x) &= \exp(\tfrac{1}{2}||x_i||^2) \;/\; \sum_i \exp(\tfrac{1}{2}||x_i||^2) \\
P(q|x_i) &= \mathcal{N}(q\,;\, \mu{=}x_i, \sigma{=}1)
\end{aligned}
$$

With this interpretation of $x$, we can use the fact that $P(i|x,q) \propto P(i|x)\, P(q|x_i) \propto \exp(q{\cdot}x_i)$ (ignoring factors independent of $i$) to reinterpret a standard attention function.

Since Transformer has a discrete segmentation of its representation into positions (which we call entities), but no explicit representation of structure, we can think of this representation as a bag of vectors (BoV, i.e. a set of instances of vectors). Each layer has a BoV representation, which is aligned with the BoV representation below it. The final output only becomes a sequence if the downstream task imposes explicit sequential structure on it, which attention alone does not.

These bag of vector representations have two very interesting properties for natural language. First, the number of vectors in the bag can grow arbitrarily large, which captures the unbounded nature of language. Secondly, the vectors in the bag are *exchangeable*, in the sense of Jordan (2010). In other words, renumbering the indices used to refer to the different vectors will not change the interpretation of the representation.[3] This is because the learned parameters in Transformer are shared across all positions. These two properties are clearly related; exchangeability allows learning to generalise to unbounded representations, since there is no need to learn about indices which are not in the training data.

These properties mean that BoV representations are nonparametric representations. In other words, the specification of a BoV representation cannot be done just by choosing values for a fixed set of parameters. The number of parameters you need grows with the size of the bag. This is crucial for language because the amount of information conveyed by a text grows with the length of the text, so we need nonparametric representations.

To illustrate the usefulness of this view of BoVs as nonparametric representations, we propose to use methods from Bayesian learning to define a prior distribution over BoVs where the size of the bag is not known. Such a prior would be needed for learning the number of entities in a Transformer representation, discussed below, using variational Bayesian approaches. For this example, we will use the above interpretation of a BoV $x{=}\{x_i \,|\, 1{\leq}i{\leq}k\}$ as specifying a distribution over queries, $P(q|x){=}\sum_i P(i|x)P(q|x_i)$. A prior distribution over these $P(q|x)$ distributions can be

[3] These indices should not be confused with position embeddings. In fact, position embeddings are needed precisely because the indices are meaningless to the model.

specified, for example, with a Dirichlet Process, $DP(\alpha, G_0)$. The concentration parameter $\alpha$ controls the generation of a sequence of probabilities $\rho_1, \rho_2, \ldots$, which correspond to the $P(i|x)$ distribution (parameterised by the $||x_i||$). The base distribution $G_0$ controls the generation of the $P(q|x_i)$ distributions (parameterised by the $x_i$).

The use of exchangeability to support generalisation to unbounded representations implies a third interesting property, discrete segmentation into entities. In other words, the information in a BoV is spread across an integer number of vectors. A vector cannot be half included in a BoV; it is either included or not. In changing from a vector space to a bag-of-vector space, the only change is this discrete segmentation into entities. In particular, no discrete representation of structure is added to the representation. Thus, the BoV representation of Transformer is the minimal nonparametric extension of a vector space.

With this minimal nonparametric extension, Transformer is able to explicitly represent entities and their properties, and implicitly represent a structure of relations between these entities. The continuing astounding success of Transformer in natural language understanding tasks suggests that this is an adequate deep learning architecture for the kinds of structured representations needed to account for the nature of language.

## 5 Looking Forward: Inducing Levels and their Entities

As argued above, the great success of neural networks in NLP has not been because they are radically different from pre-neural computational theories of language, but because they have succeeded in replacing hand-coded components of those models with learned components which are specifically designed to capture the same generalisations. We predict that there is at least one more hand-coded aspect of these models which can be learned from data, but question whether they all can be.

Transformer can learn representations of entities and their relations, but current work (to the best of our knowledge) all assumes that the set of entities is a predefined function of the text. Given a sentence, a Transformer does not learn how many vectors it should use to represent it. The number of positions in the input sequence is given, and the number of token embeddings is the same as the number of input positions. When a Transformer decoder generates a sentence, the number of positions is chosen by the model, but it is simply trying to guess the number of positions that would have been given if this was a training example. These Transformer models never try to induce the number of token embeddings they use in an unsupervised way.[4]

Given that current models hard-code different token definitions for different tasks (e.g. character embeddings versus word embeddings versus sentence embeddings), it is natural to ask whether a specification of the set of entities at a given level of representation can be learned. There are models which induce the set of entities in an input text, but these are (to the best of our knowledge) not learned jointly with a downstream deep learning model. Common examples include BPE (Sennrich et al., 2016) and unigram language model (Kudo, 2018), which use statistics of character n-grams to decide how to split words into subwords. The resulting subwords then become the entities for a deep learning model, such as Transformer (e.g. BERT), but they do not explicitly optimise the performance of this downstream model. In a more linguistically-informed approach to the same problem, statistical models have been proposed for morphology induction (e.g. (Elsner et al., 2013)). Also, Semi-Markov CRF models (Sarawagi and Cohen, 2005) can learn segmentations of an input string, which have been used in the output layers of neural models (e.g. (Kong et al., 2015)). The success of these models in finding useful segmentations of characters into subwords suggests that learning the set of entities can be integrated into a deep learning model. But this task is complicated by the inherently discrete nature of the segmentation into entities. It remains to find effective neural architectures for learning the set of entities jointly with the rest of the neural model, and for generalising such methods from the level of character strings to higher levels of representation.

The other remaining hand-coded component of computational linguistic models is levels of representation. Neural network models of language typically only represent a few levels, such as the character sequence plus the word sequence, the word sequence plus the syntax tree, or the word sequence plus the syntax tree plus the predicate-argument structure (Henderson et al., 2013; Swayamdipta

[4]Recent work on inducing sparsity in attention weights (Correia et al., 2019) effectively learns to reduce the number of entities used by individual attention heads, but not by the model as a whole.

et al., 2016). And these levels and their entities are defined before training starts, either in preprocessing or in annotated data. If we had methods for inducing the set of entities at a given level (discussed above), then we could begin to ask whether we can induce the levels themselves.

One common approach to inducing levels of representation in neural models is to deny it is a problem. Seq2seq and end2end models typically take this approach. These models only include representations at a lower level, both for input and output, and try to achieve equivalent performance to models which postulate some higher level of representation (e.g. (Collobert and Weston, 2008; Collobert et al., 2011; Sutskever et al., 2014; Vinyals et al., 2015)). The most successful example of this approach has been neural machine translation. The ability of neural networks to learn such models is impressive, but the challenge of general natural language understanding is much greater than machine translation. Nonetheless, models which do not explicitly model levels of representation can show that they have learned about different levels implicitly (Peters et al., 2018; Tenney et al., 2019).

We think that it is far more likely that we will be able to design neural architectures which induce multiple levels of representation than it is that we can ignore this problem entirely. However, it is not at all clear that even this will be possible. Unlike the components previously learned, no linguistic theory postulates different levels of representation for different languages. Generally speaking, there is a consensus that the levels minimally include phonology, morphology, syntactic structure, predicate-argument structure, and discourse structure. This language-universal nature of levels of representation suggests that in humans the levels of linguistic representation are innate. This draws into question whether levels of representation can be learned at all. Perhaps they are innate because human brains are not able to learn them from data. If so, perhaps it is the same for neural networks, and so attempts to induce levels of representation are doomed to failure.

Or perhaps we can find new neural network architectures which are even more powerful than what is now thought possible. It wouldn't be the first time!

## 6 Conclusions

We conclude that the nature of language has influenced the design of deep learning architectures in fundamental ways. Vector space representations (as in MLPs) are not adequate, nor are vector spaces which evolve over time (as in LSTMs). Attention-based models are fundamentally different because they use bag-of-vector representations. BoV representations are nonparametric representations, in that the number of vectors in the bag can grow arbitrarily large, and these vectors are exchangeable.

With BoV representations, attention-based neural network models like Transformer can model the kinds of unbounded structured representations that computational linguists have found to be necessary to capture the generalisations in natural language. And deep learning allows many aspects of these structured representations to be learned from data.

However, successful deep learning architectures for natural language currently still have many hand-coded aspects. The levels of representation are hand-coded, based on linguistic theory or available resources. Often deep learning models only address one level at a time, whereas a full model would involve levels ranging from the perceptual input to logical reasoning. Even within a given level, the set of entities is a pre-defined function of the text.

This analysis suggests that an important next step in deep learning architectures for natural language understanding will be the induction of entities. It is not clear what advances in deep learning methods will be necessary to improve over our current fixed entity definitions, nor whether the resulting entities will be any different from the ones postulated by linguistic theory. If we can induce the entities at a given level, a more challenging task will be the induction of the levels themselves. The presumably-innate nature of linguistic levels suggests that this might not even be possible.

But of one thing we can be certain: the immense success of adapting deep learning architectures to fit with our computational-linguistic understanding of the nature of language will doubtless continue, with greater insights for both natural language processing and machine learning.

## Acknowledgements

We would like to thank Paola Merlo, Suzanne Stevenson, Ivan Titov, members of the Idiap NLU group, and the anonymous reviewers for their comments and suggestions.

## References


Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2001. A neural probabilistic language model. In *Advances in Neural Information Processing Systems 13*, pages 932–938. MIT Press.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Machine Learning Research*, 3:1137–1155.

Curt Burgess. 1998. From simple associations to the building blocks of language: Modeling meaning in memory with the HAL model. *Behavior Research Methods, Instruments, & Computers*, 30(2):188–198.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proc. 14th National Conference on Artificial Intelligence*, Providence, RI. AAAI Press/MIT Press.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Noam Chomsky. 1959. On certain formal properties of grammars. *Information and Control*, 2:137–167.

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proc. 35th Meeting of Association for Computational Linguistics and 8th Conf. of European Chapter of Association for Computational Linguistics*, pages 16–23, Somerset, New Jersey.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the Twenty-Fifth International Conference (ICML 2008)*, pages 160–167, Helsinki, Finland.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184, Hong Kong, China. Association for Computational Linguistics.

Fabrizio Costa, Vincenzo Lombardo, Paolo Frasconi, and Giovanni Soda. 2001. Wide coverage incremental parsing by learning attachment preferences. pages 297–307.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

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

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734. ICLR 2017.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–212.

Jeffrey L. Elman. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7:195–225.

Micha Elsner, Sharon Goldwater, Naomi Feldman, and Frank Wood. 2013. A joint learning model of word segmentation, lexical acquisition, and phonetic variability. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 42–54, Seattle, Washington, USA. Association for Computational Linguistics.

Katrin Erk. 2010. What is word meaning, really? (and how can distributional models help us describe it?).

In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, pages 17–26, Uppsala, Sweden. Association for Computational Linguistics.

Jerry A. Fodor and B. McLaughlin. 1990. Connectionism and the problem of systematicity: Why smolensky's solution doesn't work. *Cognition*, 35:183–204.

Jerry A. Fodor and Zenon W. Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71.

James Henderson. 1994. *Description Based Parsing in a Connectionist Network*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA. Technical Report MS-CIS-94-46.

James Henderson. 1996. A connectionist architecture with inherent systematicity. In *Proceedings of the Eighteenth Conference of the Cognitive Science Society*, pages 574–579, La Jolla, CA.

James Henderson. 2000. Constituency, context, and connectionism in syntactic parsing. In Matthew Crocker, Martin Pickering, and Charles Clifton, editors, *Architectures and Mechanisms for Language Processing*, pages 189–209. Cambridge University Press, Cambridge UK.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proc. joint meeting of North American Chapter of the Association for Computational Linguistics and the Human Language Technology Conf.*, pages 103–110, Edmonton, Canada.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 95–102, Barcelona, Spain.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):949–998.

E.K.S. Ho and L.W. Chan. 1999. How to design a connectionist holistic parser. *Neural Computation*, 11(8):1995–2016.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

K. Hornik, M. Stinchcombe, and H. White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366.

Ajay N. Jain. 1991. *PARSEC: A Connectionist Learning Architecture for Parsing Spoken Language*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

M.I. Jordan. 2010. Bayesian nonparametric learning: Expressive priors for intelligent systems. In R. Dechter, H. Geffner, and J. Halpern, editors, *Heuristics, Probability and Causality: A Tribute to Judea Pearl*, chapter 10. College Publications.

Aravind K. Joshi. 1987. An introduction to tree adjoining grammars. In Alexis Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam.

Aravind K. Joshi, K. Vijay-Shanker, and David Weir. 1990. The convergence of mildly context-sensitive grammatical formalisms. In Peter Sells, Stuart Shieber, and Tom Wasow, editors, *Foundational Issues in Natural Language Processing*. MIT Press, Cambridge MA. Forthcoming.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing universal dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2015. Segmental recurrent neural networks.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Yann LeCun and Yoshua Bengio. 1995. Convolutional networks for images, speech, and time-series. In Michael A. Arbib, editor, *The handbook of brain theory and neural networks (Second ed.)*, page 276–278. MIT press.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

C. von der Malsburg. 1981. The correlation theory of brain function. Technical Report 81-2, Max-Planck-Institute for Biophysical Chemistry, Gottingen.

Diego Marcheggiani, Joost Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 486–492, New Orleans,

Louisiana. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

Risto Miikkulainen. 1993. *Subsymbolic Natural Language Processing: An integrated model of scripts, lexicon, and memory*. MIT Press, Cambridge, MA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Alireza Mohammadshahi and James Henderson. 2019. Graph-to-graph transformer for transition-based dependency parsing.

Alireza Mohammadshahi and James Henderson. 2020. Recursive non-autoregressive graph-to-graph transformer for dependency parsing with iterative refinement.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

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

Carl Pollard and Ivan A. Sag. 1987. *Information-Based Syntax and Semantics. Vol 1: Fundamentals*. Center for the Study of Language and Information, Stanford, CA.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–175.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of bert. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8594–8603. Curran Associates, Inc.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986a. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing, Vol 1*, pages 318–362. MIT Press, Cambridge, MA.

D. E. Rumelhart, J. L. McClelland, and the PDP Reseach group. 1986b. *Parallel Distributed Processing: Explorations in the microstructure of cognition, Vol 1*. MIT Press, Cambridge, MA.

Sunita Sarawagi and William W Cohen. 2005. Semi-markov conditional random fields for information extraction. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1185–1192. MIT Press.

Hinrich Schütze. 1993. Word space. In *Advances in Neural Information Processing Systems 5*, pages 895–902. Morgan Kaufmann.

Mark S. Seidenberg. 2007. Connectionist models of reading. In Gareth Gaskell, editor, *Oxford Handbook of Psycholinguistics*, chapter 14, pages 235–250. Oxford University Press.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Lokendra Shastri and Venkat Ajjanagadde. 1993. From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*, 16:417–451.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Paul Smolensky. 1988. On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1–17.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013a. Parsing with compositional vector grammars. In *Proceedings of the*

*51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack LSTMs. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 187–197, Berlin, Germany. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ivan Titov and James Henderson. 2007a. A latent variable model for generative dependency parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 144–155, Prague, Czech Republic. Association for Computational Linguistics.

Ivan Titov and James Henderson. 2007b. A latent variable model for generative dependency parsing. In *Proceedings of the International Conference on Parsing Technologies (IWPT'07)*, Prague, Czech Republic. Association for Computational Linguistics.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2773–2781. Curran Associates, Inc.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Majid Yazdani and James Henderson. 2015. Incremental recurrent neural network dependency parser with search-based discriminative training. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 142–152, Beijing, China. Association for Computational Linguistics.