Evaluating Transformer's Ability to Learn Mildly Context-Sensitive Languages

Shunjie Wang

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2021

Committee:

Shane Steinert-Threlkeld

Noah A. Smith

Program Authorized to Offer Degree:
Linguistics

University of Washington

Abstract

Evaluating Transformer's Ability to Learn
Mildly Context-Sensitive Languages

Shunjie Wang

Chair of the Supervisory Committee:
Shane Steinert-Threlkeld
Department of Linguistics

Transformer models perform well on NLP tasks, but recent theoretical studies suggest their ability in modeling certain regular and context-free languages are limited. This creates a disparity given their success in modeling natural language strings, which are hypothesized to be mildly context-sensitive. We complement previous works on transformers and formal languages by relating them to mildly context-sensitive grammar formalisms with varying degrees of weak generative capacity. We test simple vanilla transformer models' ability to learn copying, crossing, and multiple agreements languages, and find that they generalize well to unseen in-domain data and have comparable performance to LSTMs, and learn highly interpretable self-attention patterns. However, such transformers cannot consistently recognize strings from the languages that are longer than the ones seen during training, and are often outperformed by LSTMs in this setting. We present initial evidence that suggests this is due to the limitation of the vanilla sinusoidal positional encoding.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I would like to first thank my advisor Prof. Shane Steinert-Threlkeld for introducing me to neural language models through his seminar and advising me on this project. This thesis is motivated by a slightly ambitious attempt of mine to synthesize parts of formal language theory, theoretical linguistics, and current advances in NLP. There are a million things that look interesting to me at the intersection of the three, and Shane helped me to narrow down the scope and converge the scattered interests into a unifying theme that is being presented here. I am very fortunate to have Shane advising me as he is knowledgeable and experienced in both classical formal languages and automata theory as well as current methods in deep learning.

I would also like to thank my reader Prof. Noah Smith from Computer Science and Engineering for reading the thesis and giving detailed feedback and helpful comments and pointers. I am humbled after completing this work, understanding that there are a lot more things for me to learn on this topic, but nevertheless received encouraging and generous comments from both Shane and Noah, for which I am immensely grateful.

Lastly, I would like to thank my friends and bright young researchers Wang, Yang (UCLA Linguistics) and Zhou, Pei (USC CS) for helping to read the draft of this thesis. Any remaining errors are my own.

DEDICATION

献给我的家人

to my family

Chapter 1

INTRODUCTION

Transformer models (Vaswani et al., 2017) have made tremendous accomplishments in Natural Language Processing (NLP) in recent years, from their initial contribution to neural machine translation, to recent advances in pre-trained neural language models such as BERT (Devlin et al., 2019), and all the advances in the downstream tasks benefited by these models. They improved on previous neural sequence models such as Long Short-Term Memory (LSTM; Hochreiter and Schmidhuber, 1997) or Gated Recurrent Units (GRU; Cho et al., 2014) without using recurrence, which is prone to the problem of vanishing gradients in absence of additional techniques like attention mechanism. Instead, transformers rely on a flavor of attention mechanism called self-attention, which allows the parallel processing of individual tokens in the sequence.

Despite the improvements, there is a disparity between its formal theoretical capacity and its practical success. Hahn (2020) found that hard-attention transformers are theoretically limited, since they cannot even recognize some regular and context-free languages when analyzed asymptotically. The results are also applicable to soft-attention transformers under certain assumptions. This is counterintuitive given its success in NLP tasks, which motivated us to ask that what languages can the transformer actually recognize, and looking at its practical ability using empirical methods.

However, answering what the set of languages that transformer accepts is could be a very difficult task, since we do not even quite know yet what set of languages do LSTMs recognize. Therefore, we are approaching the problem by relating it to grammar formalisms with different weak generative capacities in theoretical linguistics, and to see how the transformer's capacity compares to them. Previous work such as Weiss et al. (2018); Suzgun

et al. (2019); Bhattamishra et al. (2020a) on neural sequence models and formal languages have studied a variety of regular and context-free languages. However, this work looks at a more linguistically significant class. Specifically, we seek to study a formal class of grammars and languages that is hypothesized to have the necessary expressive power to describe the complexity of natural language, the mildly context-sensitive class.

This thesis asks two questions:

1. Can transformer learn to recognize mildly context-sensitive languages from training on finite examples? How well does it generalize to sequences with similar lengths as the ones seen during training, as well as unseen, longer sequences?

2. In addition to solving the task, what internal mechanisms and representations do transformers learn to recognize these languages?

The rest of this thesis is organized as follows: we introduce the necessary background on mildly context-sensitive grammars and languages as well as the transformer architecture in Chapter 2. Chapter 3 reviews some recent work on relating neural sequence models and formal language recognition, and summarizes the current results on specific classes and families of languages. Chapter 4 explains the two task setups to be used, as well as the particular sets of languages studied. We present the results and visualizations in Chapter 5, and discuss the insights we get from them in Chapter 6, as well as taking a step back to ask how powerful a model needs to be for modeling natural language. Finally, we wrap up in Chapter 7 and point to a number of possible extensions to this work.

Chapter 2
BACKGROUND

This chapter lays an appropriate background for the problem discussed. The first half briefly presents how formal language theory has been used to study natural languages, as well as a hypothesis of the formal complexity of natural language, which motivates the class of languages to be studied in this thesis. Then, we briefly go over the architecture of the transformer model, especially the encoder part, as well as the necessary background on attention mechanism.

2.1 Natural Language as Formal Language

2.1.1 Chomsky Hierarchy

Following Chomsky (1956, 1957), we regard a language L as a set of finite-length sentences made of a finite set of elements. A grammar of L is a device that generates all grammatical sentences in L, and rejects all ungrammatical ones. Such a device is based on finitely many observed sentences, then generalizes to infinite grammatical sentences by establishing grammatical rules.

In the formal sense, a grammar G is a 4-tuple (V, Σ, R, S), where V is a set of non-terminal symbols; Σ is a set of terminal symbols, or an alphabet; R is a set of rewriting rules of the form $\varphi \to \psi$, where φ, ψ are strings, and $S \in V$ is the starting symbol.

An unrestricted grammar has rewrite rules of form $\alpha \to \beta$, where α, β are strings of any non-terminal and terminal symbols, but α cannot be the empty string. This grammar is too powerful for natural language in terms of weak generative capacity, that is, the set of strings it is capable of generating. An unrestricted grammar has the computational power equivalent to arbitrary Turing machines, the general purpose abstract machine that can perform any

set of operations that count as a computation (Partee et al., 1993). Thus, such a grammar is capable of generating way more languages than all possible natural languages. Secondly, while an unrestricted grammar can indeed compute natural language strings, it does not guarantee meaningful structural descriptions such as a tree structure for the sentence and therefore lacks linguistic interests (Chomsky, 1959). On the other hand, a computational device that is (much) less powerful than a Turing machine would be a finite-state automaton, which as Chomsky (1956) argues, is too weak to model English, in the sense that it cannot generate all and only grammatical English sentences.[1] Thus, what intermediate systems lie between these two extremities, such that they have the proper generative power for human language, and assign structural descriptions to natural language sentences, is of interest here.

To get these intermediate systems starting from the most powerful Turing machines, one would impose increasingly heavy restrictions on what languages can be generated, until reaching the most restricted finite-state automata. Chomsky (1959) discussed three such increasingly heavy restrictions: Restriction 1 requires the rewrite rules to be of form $\varphi_1 A \varphi_2 \rightarrow \varphi_1 \omega \varphi_2$, where A is a single non-terminal symbol; $\omega, \varphi_1, \varphi_2$ are arbitrary strings of non-terminals and terminals, except that ω is non-empty. This is saying that A can be rewritten as ω in the surrounding environment of $\varphi_1 \_ \varphi_2$. A grammar meeting Restriction 1 is called Type 1 grammar, and the language it generates is Type 1 language. In Chomsky (1963), these are also called context-sensitive grammar and language (henceforth CSG/CSL).

Restriction 2 requires the rewrite rules to be of form $A \rightarrow \omega$. This is very similar to Restriction 1 except that the surrounding environment is now null. Thus, rules satisfying Restriction 2 also satisfy Restriction 1. The corresponding grammar and language are Type 2, which are also known as context-free grammar and language (henceforth CFG/CFL) in Chomsky (1963).

Lastly, Restriction 3 requires the rewrite rules to be of form $A \rightarrow aB$ or $A \rightarrow a$, where a is a terminal symbol, and B is a non-terminal symbol. Since ω in Restriction 2 can well

[1] Although this argument is well-accepted, the proof used in Chomsky (1956, 1957) is mathematically flawed (Pullum, 2007). We briefly elaborate on this towards the end of Section 2.1.2.

Type	Grammar	Language	Rewriting Rule
0	Unrestricted Grammar	Recursively Enumerable Language	$\alpha \to \beta$
1	Context-Sensitive Grammar	Context-Sensitive Language	$\varphi_1 A \varphi_2 \to \varphi_1 \omega \varphi_2$
2	Context-Free Grammar	Context-Free Language	$A \to \omega$
3	Regular Grammar	Regular Language	$A \to aB \mid a$

Table 2.1: Type 0, 1, 2, and 3 grammar and language

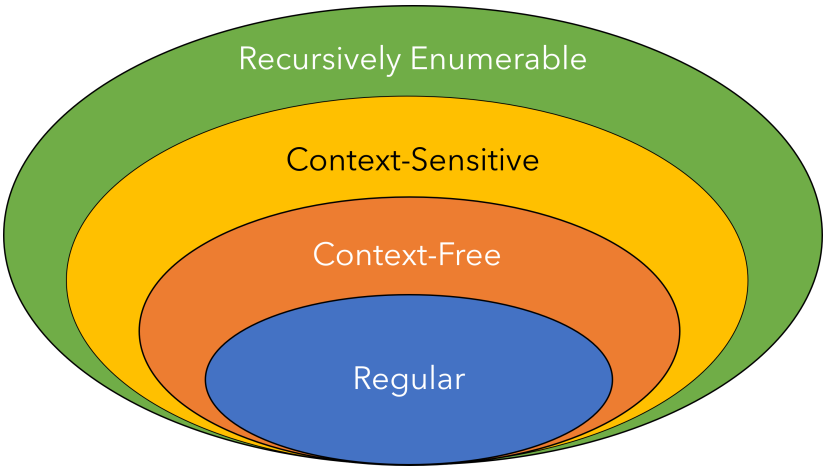
be aB or a, satisfying Restriction 3 also gives 2, and in turn 1. The corresponding grammar and language are Type 3, and this is exactly the class of languages that finite-state machines can recognize.

Type 0 grammars are unrestricted, which are essentially Turing machines. Every Type 0 language is a recursively enumerable set of strings and vice versa. Table 2.1 summarizes the above defined grammars and languages.

By the definition above, for both grammars and languages, we have (Chomsky, 1959):

$$\text{Type } 3 \subseteq \text{Type } 2 \subseteq \text{Type } 1 \subseteq \text{Type } 0$$

This is now commonly known as the Chomsky hierarchy, a hierarchy of classes of formal languages, which can also be expressed as the following Venn diagram:



Then, where does human language fit on this hierarchy becomes an interesting question. Chomsky (1956) discussed context-free phrase structure grammar as a model for natural language grammar, but cannot prove whether English or some other language is beyond the descriptive power of context-free phrase structures. He still argued the inadequacy of context-free phrase structure grammar for not being reasonably simple as a language description, but nevertheless left the generative capacity question open.

A more thorough discussion of Chomsky hierarchy can be found in Hunter (2021).

2.1.2 Non-Context-Freeness of Natural Languages

Following Chomsky's posing of the open question of whether a context-free grammar is sufficient for English and other natural languages alike, there have been a number of attempts to prove the negative. However, most of these claims were considered invalid as either the proof is mathematically flawed, or the linguistic data cited is empirically problematic (Pullum and Gazdar, 1982). A very close argument is the cross-serial dependency in Dutch subordinate clauses (Bresnan et al., 1982). However, an alternative context-free analysis of the Dutch phenomenon can be given (for example, in Pullum and Gazdar (1982)).

The first widely accepted proof is made by Shieber (1985) on the non-context-freeness of Swiss German (*Schwytzerdütsch*). This argument is very similar to the Dutch one above since Swiss German is also a Germanic language and also has cross-serial dependencies in subordinate clauses, except that Dutch does not have explicit case marking but Swiss German does, thus an alternative context-free analysis can be used to challenge the Dutch argument but not Swiss German.

The cross-serial dependency in Swiss German subordinate clauses is illustrated by the following example:

(1) *Jan säit das mer em Hans es huus hälfe aastriiche*
Jan says that we Hans-DAT the house-ACC help paint
'Jan says that we help Hans paint the house.'

Arcs were drawn to connect the dependents, and it can be clearly seen that the arcs are crossing each other. The robustness of the phenomenon is demonstrated by the following triply crossing embedded clause, which is grammatical when having the correct case marking:

(2) *Jan säit das mer d'chind em Hans es huus lönd hälfe aastriiche*
Jan says that we the children-ACC Hans-DAT the house-ACC let help paint
'Jan says that we let the children help Hans paint the house.'

To prove how this makes Swiss German non-CF, for some language L that satisfies the properties of Swiss German, e.g., the matching of order and number of correspondingly cased verbs and objects in subordinate clauses, we define the following homomorphism for such L:

$$f(\omega) = \begin{cases} \mathtt{w} & \omega = \text{“Jan säit das mer”} \\ \mathtt{a} & \omega = \text{“d'chind”} \\ \mathtt{b} & \omega = \text{“em Hans”} \\ \mathtt{x} & \omega = \text{“es huus”} \\ \mathtt{c} & \omega = \text{“lönd”} \\ \mathtt{d} & \omega = \text{“halfe”} \\ \mathtt{y} & \omega = \text{“astriiche”} \\ \mathtt{z} & \text{otherwise} \end{cases}$$

We intersect $f(L)$ with a regular language $\mathtt{wa}^*\mathtt{b}^*\mathtt{xc}^*\mathtt{d}^*\mathtt{y}$, and we get language $\mathtt{wa}^n\mathtt{b}^m\mathtt{xc}^n\mathtt{d}^m\mathtt{y}$. We can show that $\mathtt{wa}^n\mathtt{b}^m\mathtt{xc}^n\mathtt{d}^m\mathtt{y}$ is non-context-free via pumping lemma, or by taking an additional homomorphic image to yield $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$, a typical non-context-free language. Since CFLs are closed under homomorphism and intersection with regular languages, we thus know Swiss German must not be a CFL.

Another example that survived scrutiny was reduplication in Bambara (Mande, Mali) by Culy (1985). Bambara has Noun *o* Noun constructions such that when intersecting with a regular language yield a language of the form $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$, and by the same reasoning as above, it shows that the vocabulary of Bambara is non-context-free.

A common misconception to clarify here is that the presence of a non-context-free construction itself like cross-serial dependency in a natural language, implies nothing about whether the natural language as a whole is also non-context-free (Partee et al., 1993; Mohri and Sproat, 2006). In fact, cross-serial constructions are not rare cross-linguistically (cf. Stabler (2004)), but there is a reason why almost only Swiss German is cited when discussing the non-context-freeness of natural language. A lot of work attempted to discuss the computational complexity of a particular construction, but then overgeneralized that conclusion to the entire language, which is flawed (Mohri and Sproat, 2006). For example, although it is well accepted that natural language is supra-regular, Pullum (2007) indicates that the proof Chomsky used for showing the unsuitability of finite-state model for English in *Syntactic Structures* (Chomsky, 1957) is flawed. Chomsky argues that English contains several non-finite-state models, but subparts of a natural language demonstrating some properties does not entail anything about the language in its entirety. A proper proof should extract a subset of a language via intersection with a regular set, and then use the closure properties of intersection with regular sets to make arguments about the complexity of the language. This is exactly the technique used to prove non-context-freeness of Swiss German and Bambara above.

Some good reviews on computational complexity and formal language theory for natural language include Jäger and Rogers (2012); Branco (2018).

2.1.3 Mildly Context-Sensitive Hypothesis

Despite proving that the complexity of natural language is supra-context-free, the next formal class on the Chomsky hierarchy, the context-sensitive class, is undesired for modeling natural language strings, as it still has much more generative capacity than natural language ever

needs. To illustrate that, one can even design a context-sensitive grammar for a language as complex as the set of strings whose length is a prime number[2] (Jäger and Rogers, 2012; Hunter, 2021). The inefficiency is more problematic for computational linguists: context-sensitive language parsing cannot be performed in polynomial time.

Joshi (1985) has proposed Tree Adjoining Grammar (TAG) such that extends CFG, but only adds the necessary amount of context-sensitivity for treating the non-context-free linguistic phenomena, characterized by the following three properties of the formalism:

1. Limited cross-serial dependencies: only a limited number of cross-serial dependencies are allowed by TAG, e.g., cross-serial dependencies in Dutch subordinate clauses, but not the ones in MIX $= \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid |w|_{\mathtt{a}} = |w|_{\mathtt{b}} = |w|_{\mathtt{c}}\}$, i.e., the language of strings with the same number of a's, b's, and c's but the symbols can occur in any order.

2. Constant growth: when strings of a language are arranged in increasing order by their lengths, any two consecutive lengths do not differ by arbitrarily large amounts (Joshi, 1987). Thus, languages like $\mathtt{a}^{2^n}$ or $\mathtt{a}^{n^2}$ are not in TAG because of the exponential and quadratic growth respectively.

3. Polynomial parsing: TAGs can be parsed in $O(n^4)$ time.

Such extra power is not gained from ad hoc modification of CFG, but is a result of how the formalism handles linguistic phenomena like recursion and dependencies.

The three properties roughly categorize a class of grammars and languages that he calls *mildly context-sensitive grammars/languages* (henthforth MCSG/MCSL). Although these characterizations were derived from the properties of TAG, thus biased towards the formalism, a few other formalisms, such as Combinatory Categorial Grammar (CCG; Steedman 1987), also count as MCSG by these properties. It was even shown that CCG is weakly equivalent to TAG (Vijay-Shanker and Weir, 1994), that is, with $L(G)$ denoting the set of languages defined by the grammar G, we have $L(\text{CCG}) = L(\text{TAG})$.

---

[2]This follows from that the automaton recognizes context-sensitive languages is a Linear Bounded Automaton (LBA), and an LBA can recognize the set of primes (Hartmanis and Shank, 1968).

These rough properties were later furthur formalized to define the set of mildly context-sensitive languages. By the definition in Kallmeyer (2010), reproduced as the following with adaptations, a set of languages $\mathcal{L}$ is mildly context-sensitive if and only if:

1. Limited cross-serial dependencies: there is an $n \geq 2$ such that for all $k \leq n$, language $\{w^k \mid w \in \Sigma^*\}$ is in the set $\mathcal{L}$.
2. Constant growth: for some language $L \in \mathcal{L}$, there exists some constant $c_0 > 0$ and a finite set of constants $C \subset \mathbb{N} - \{0\}$ such that for all $w \in L$, $|w| > c_0$, there is a $w' \in L$ with $|w| = |w'| + c, c \in C$ (Weir, 1988).
3. Polynomial parsing: $\mathcal{L} \in \mathsf{PTIME}$.
4. $\mathcal{L}$ contains all context-free languages.

Despite the above formal characterization for MCS languages, there is no single grammar formalism that generates the largest possible set of such languages. The closest approximation we have is Linear Context Free Rewriting Systems (LCFRS) proposed by Vijay-Shanker et al. (1987), and equivalently Multiple Context Free Grammar (MCFG) by Seki et al. (1991).

Stabler (1997) proposed Minimalist Grammar (MG) as a formalization of Chomsky's latest attempt to syntactic theory, the Minimalist Program. He argued that it is more expressive than TAG and equivalents. For example, it can handle a language like $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$ that has more than 4 counting dependencies, while TAG can only recognize up to $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$ which has exactly 4 counting dependencies. The need for such additional power is motivated by linguistic phenomenon like German scrambling (Becker et al., 1991). MG is proved to be mildly context-sensitive by showing its weak equivalence to MCFG.[3] Independently, Michaelis (1998) also established the mild context-sensitivity of MG by showing its weak equivalence to LCFRS.

We can write the following hierarchy for the above formalisms based on the languages they generate (Stabler, 2011):

$$\text{CFL} \subset \boxed{L(\text{TAG}) = L(\text{CCG})} \subset \boxed{L(\text{MG}) = L(\text{LCFRS}) = L(\text{MCFG})} \subset \text{CSL}$$

[3]This follows from $L(\text{MG}) \subseteq L(\text{MCFG})$ (Michaelis, 1998) and $L(\text{MCFG}) \subseteq L(\text{MG})$ (Harkema, 2001) respectively.

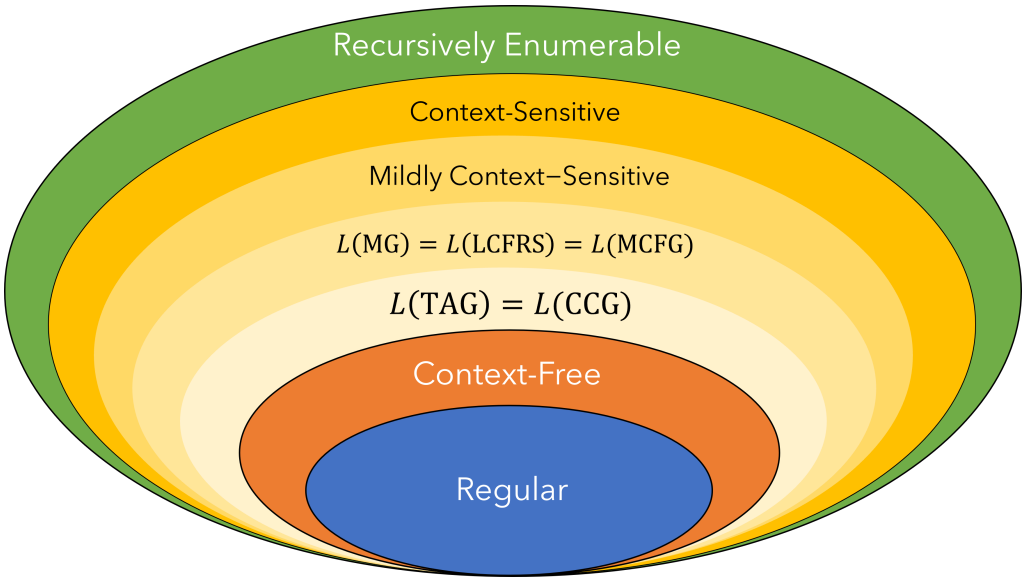
A hypothesis for the complexity of natural language is that it is mildly context-sensitive. Whether the hypothesis is true remains an open question. There have been a few works that discuss potential challenges to the MCS hypothesis, and mainly attack whether constant growth should be a property of natural language (Kallmeyer, 2010). The earlier challenges to the MCS hypothesis were already refuted, such as Old Georgian case stacking in Michaelis and Kracht (1997), which is both refuted by Bhatt and Joshi (2004) and the robustness of the phenomenon cannot be verified since the language is dead; and Chinese number names in Radzinski (1991), which is based on questionable data so rare in naturally occuring sentences that they may not be part of Chinese syntax. A recent study on copying in Yoruba (Niger-Congo, West Africa; Kobele 2006) argues it requires power beyond that of MCS grammar such that allows copies of copies. To model Yoruba copying, Kobele (2006) extended standard MG with so-called copy movement, and the resulting extension has power beyond MCFG, and recognizes the exponential language $\mathtt{a}^{2^n}$, which violated the constant growth property of mild context-sensitivity. Graf (2021) argues that there is no known challenge to the Yoruba argument at this time[4], but there is no consensus on whether such additional power is truly needed for natural language syntax either.

Nevertheless, mild context-sensitivity is the basis for the design of formalisms like TAG or CCG. Kobele also noted that the Yoruba case does not suggest the formal complexity of natural language is unconstrained, since efficient parsing remains a desired condition. Given these counterarguments are inconclusive, it is still reasonably good to *assume* that natural language is mildly context-sensitive (Kallmeyer, 2010; Jäger and Rogers, 2012), acknowledging that what the exact class is should remain an open question.

For the remainder of this thesis, we refer to the languages recognizable by TAG and its equivalents as $L(\text{TAG})$, and languages recognizable by MG and its equivalents as $L(\text{MG})$. $L(\text{TAG})$ and $L(\text{MG})$ are thus language complexity classes that we are going to look at in

[4]However, it is hard to verify to what extent researchers outside of the MG community are aware of the Yoruba argument, e.g., Kallmeyer (2010); Jäger and Rogers (2012) did not cite the Yoruba challenge when conjecturing NL to be MCS or MG-recognizable.

Chapter 4. To demonstrate the hierarchy of grammars discussed so far, the following Venn diagram as an extension to the original Chomsky hierarchy can be compiled:



2.2 Attention, Self-Attention and Transformers

2.2.1 RNN Encoder-Decoder Architecture and Attention Mechanism

Attention mechanism is originally proposed by Bahdanau et al. (2015) for improving neural machine translation by jointly learning alignment and translation. Neural machine translation is a typical sequence-to-sequence (seq2seq) task.

Following the definition in Cho et al. (2014), a seq2seq architecture is an encoder-decoder model, where the encoder encodes an input sequence of vectors $X = \langle \vec{x}_1, \ldots, \vec{x}_{|X|} \rangle$ into some context vectors, and then the decoder decodes the context vector into the output sequence $Y = \langle \vec{y}_1, \ldots, \vec{y}_{|Y|} \rangle$.

In the encoder, at each time step $j \in \{1, \ldots, |X|\}$, a non-linear activation f, such as an LSTM unit, takes two inputs, namely the previous hidden state at last time step $\vec{h}_{j-1}$ and the current input word $\vec{x}_j$, to compute the new hidden state:

$$\vec{h}_j = f(\vec{h}_{j-1}, \vec{x}_j)$$

The encoder produces a fixed-length vector that we call context vector $\vec{c}$, which intuitively should have encoded every word in the input sentence. An option is to use the final hidden state $\vec{h}_{|X|}$:

$$\vec{c} = \vec{h}_{|X|}$$

As for the decoder, it predicts word $\vec{y}_i$ at each time step $i \in \{1, \ldots, |Y|\}$, given current decoder hidden state $\vec{s}_i$. Both $\vec{s}_i$ and $\vec{y}_i$ are also conditioned on the previous prediction $\vec{y}_{t-1}$, as well the context vector $\vec{c}$ produced by the encoder.

$$\vec{s}_t = f(\vec{s}_{t-1}, \vec{y}_{t-1}, \vec{c})$$

A problem with this approach is that since our context vector $\vec{c}$ is only the final encoder hidden state $\vec{h}_{|X|}$, it is possible that earlier signal is already weak at time step $|X|$. Especially since $\vec{c}$ is a fixed-length vector, this could be more problematic when the input X is long. Thus, it would be better to make use of the intermediate encoder hidden state $\vec{h}_t$, and one way to do so is to have respective context $\vec{c}_i$ at each decoding time step i, such that $\vec{c}_i$ is a weighted sum of all the encoder hidden states $\vec{h}_j$, which later will be referred to as *value* vectors:

$$\vec{c}_i = \sum_{j=1}^{|X|} \alpha_{ij} \vec{h}_j.$$

where the weight term α_{ij} is the softmax of some alignment model that scores the similarity between the previous decoder hidden state $\vec{s}_{i-1}$ and encoder hidden state $\vec{h}_j$, which later will be referred to as *query* and *key*, respectively:

$$\begin{aligned} \alpha_{ij} &= \mathsf{softmax}(\text{score}(\vec{s}_{i-1}, \vec{h}_j)) \\ &= \frac{\exp\Big(\text{score}(\vec{s}_{i-1}, \vec{h}_j)\Big)}{\sum_{k=1}^{|X|} \exp\Big(\text{score}(\vec{s}_{i-1}, \vec{h}_k)\Big)} \end{aligned}$$

The original alignment model used in Bahdanau et al. (2015) is a single-layer feed-forward neural network such that:

$$\text{score}(\vec{s}_{i-1}, \vec{h}_j) = \vec{v}_a^\top \tanh\Big(W_a \vec{s}_{i-1} + U_a \vec{h}_j\Big) \tag{2.1}$$

where $\vec{v}_a, W_a, U_a$ are weight matrices.

2.2.2 Query, Key, Value, and Self-Attention

The terms *query*, *key*, and *value* may be interpreted in a way that is analogous to their usage in information retrieval. In a typical search engine, the user input some keywords to be searched, which we call *query*. The query is matched against all *keys* in the index or database, and lastly *values* corresponding to the keys are returned.

The alignment score between a query $\vec{s}_t$ and a key $\vec{h}_j$ measures the similarity between the two vectors. There are several variants to how this can be calculated, one of which is the dot-product attention (Luong et al., 2015):

$$\text{score}(\vec{s}_t, \vec{h}_j) = \vec{s}_t^{\top} \vec{h}_j \tag{2.2}$$

which is simply the dot product of the two vectors. Dot product can be seen as measuring similarity since for some vectors $\vec{u}, \vec{v}$, we have $\vec{u}^{\top}\vec{v} = |\vec{u}||\vec{v}| \cos\theta$. We thus may interpret that vectors with similar magnitude and pointing to similar directions have high dot product value.

The scoring function used by transformers is very similar to this, except for the inclusion of an additional scaling factor $\frac{1}{\sqrt{d_k}}$:

$$\text{score}(\vec{s}_t, \vec{h}_j) = \frac{\vec{s}_t^{\top} \vec{h}_j}{\sqrt{d_k}} \tag{2.3}$$

where d_k is the dimension for both query and key vectors. This is known as the scaled dot-product attention. The purpose for the inclusion of a scaling factor is that for large d_k, the dot product will also get larger, and the softmax of large values has very small gradients close to 0, making the weight update hard.

Transformers use self-attention, which means that instead of having queries from the decoder output, and keys, values from the encoder input, all of query, key, and value vectors are from the same place, namely the output of the previous layer in the encoder. Thus, every encoder position attends to every encoder position in the previous layer. Parameter matrices W^Q, W^K, W^V unique to each layer were used to produce query, key, value (Shaw et al., 2018).

Query, key, and value vectors can be organized in matrices $Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, K \in \mathbb{R}^{d_{\text{model}} \times d_k}, V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, thus the attention calculation can be formulated as efficient matrix multiplication.

$$\text{Attention}(Q, K, V) = \mathsf{softmax}(\frac{QK^\top}{\sqrt{d_k}})V$$

2.2.3 Multi-head Attention

Transformer attention is also multi-headed, meaning the model can jointly attend to different positions from different representation subspaces. Taking BERT-base as an example, the pre-trained language model contains 12 layers and 12 heads each layer, totaling 144 heads. Clark et al. (2019); Manning et al. (2020) has found the attention patterns of some heads resemble certain known linguistic constructions. For example, Head 8-10 has direct object attend to verbs 86.8% of the time; Head 9-6 has prepositions attend to their objects 76.3% of the time. However, not all heads are fully exploited. Michel et al. (2019) has found that many heads might be pruned without harming the performance significantly.

Each head is given by:

$$\begin{aligned}\text{head}_i &= \text{Attention}(Q_i, K_i, V_i) \\ &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned}$$

where $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, are projection matrices. $d_k = d_v = \frac{d_{\text{model}}}{h}$, i.e. we slice each of Q, K, V into h slices, and calculate the scaled dot-product attention for the slices separately.

Thus, an h-headed multi-head attention is the concatenation of the h heads, multiplied by projection matrix $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

2.2.4 Transformer Encoder Forward Pass

This section illustrates how multi-headed self-attention fits in the forward pass, as well as introducing other important components used by transformer encoder forward pass to encode

some representation. The original transformer presented in Vaswani et al. (2017) is a seq2seq model. However, in work like Devlin et al. (2019), only the encoder part is used, which will also be the focus of this thesis.

Each encoder contains two sublayers, the first being the multi-head attention, and the second is a pointwise feedforward neural network. We now illustrate these components in more detail.

Positional Encoding

In the absence of recurrence, self-attention alone has no knowledge of word order in the sequence. Thus, the self-attention patterns would be identical for any arbitrary permutation of the words, making the model order-invariant (Pérez et al., 2019). Thus, absolute positional information of sinusoidal waves is added to the input embeddings. At postion *pos* and dimension i, the positional encoding is:

$$\mathrm{PE}_{(\mathrm{pos},2i)} = \sin\left(\frac{\mathrm{pos}}{10000^{\frac{2i}{d_{\mathrm{model}}}}}\right)$$
$$\mathrm{PE}_{(\mathrm{pos},2i+1)} = \cos\left(\frac{\mathrm{pos}}{10000^{\frac{2i}{d_{\mathrm{model}}}}}\right)$$

The sum of input embeddings and positional encoding is regularized using Dropout. A learned positional embeddings was also used, for example, in BERT. Many variants of positional encoding have since been proposed.

Residual Connection

Residual connection (He et al., 2016) means to add the original input X to a sublayer once again to the sublayer's output with the same size. There are two residual connections in the encoder, one to the multi-head attention sublayer output, and another to the feedforward neural network sublayer output. The sublayer output is regularized using Dropout with a rate of 0.1 before the addition.

$$\mathrm{Residual}(X) = X + \mathrm{Dropout}(\mathrm{Sublayer}(X))$$

Layer Normalization

Layer normalization (Ba et al., 2016) calculates the layer mean and standard deviation for all hidden units in the layer. For some hidden unit $\vec{h}$ from one time step of the output of the last sublayer after the residual connection is applied, the mean and standard deviation are:

$$\mu = \frac{1}{d_{\text{model}}} \sum_{i=1}^{d_{\text{model}}} h_i$$

$$\sigma = \sqrt{\frac{1}{d_{\text{model}}} \sum_{i=1}^{d_{\text{model}}} (h_i - \mu)^2}$$

we then organize all the individual mean and standard deviation into vectors $\vec{\mu}, \vec{\sigma}$.

The normalization is a z-score normalization, plus a small constant hyperparameter ε being an arbitrary small constant added to avoid division by zero as well as for numerical stability:

$$\text{LayerNorm}(\text{Residual}(X)) = \vec{\gamma} \cdot \frac{\text{Residual}(X) - \vec{\mu}}{\vec{\sigma} + \varepsilon} + \vec{\beta}$$

When ε is ignored, the normalized values will be normally distributed, i.e., $\frac{\vec{x} - \mu}{\sigma} \sim \mathcal{N}(0, 1)$. Similar to Batch Normalization (Ioffe and Szegedy, 2015), the normalized values are scaled by $\vec{\gamma}$ and shifted by $\vec{\beta}$, where $\vec{\gamma}, \vec{\beta}$ are learnable parameters.

Feedforward Neural Network

The feedforward neural network sublayer comes after the multi-headed self-attention sublayer. The purpose of this sublayer is to allow a place for non-linear activation like ReLU as well as to add model complexity. For some row vector $\vec{x}$ for individual time steps from output of self-attention sublayer after residual connection and layer normalization, we have:

$$\text{FFN}(\vec{x}) = \text{ReLU}(\vec{x}W_1 + \vec{b}_1)W_2 + \vec{b}_2$$

where input and output has dimension $d_{\text{model}} = 512$, and hidden layer has dimension $d_{\text{ff}} = 2048$, i.e., we have parameters $W_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}, W_2 \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}, \vec{b}_1 \in \mathbb{R}^{d_{\text{ff}}}, \vec{b}_2 \in \mathbb{R}^{d_{\text{model}}}$.

Residual connection is once again applied to add the original output from the last module to the FFN output, and the result is again normalized using layer normalization.

More details can be found in a recent survey of transformers by Lin et al. (2021).

Chapter 3
RELATED WORK

This chapter reviews some previous literature on analyzing neural sequence models using formal languages and automata theory. We summarize the current results on specific classes of languages, as well as the tasks used to study the expressive power and internal mechanisms.

3.1 Power of Transformers

Pérez et al. (2019) proved that hard-attention transformers with positional encoding are Turing-complete, given infinite precision for internal representations. Their formulation of transformer is different from Vaswani et al. (2017) in that they proposed hardmax instead of softmax in the self-attention equation, where hardmax for each $x_i \in \vec{x}$ is defined as:

$$f_{\text{hardmax}}(x_i) = \begin{cases} 1 & x_i \text{ is the maximum value} \\ 0 & \text{otherwise} \end{cases}$$

Thus, for some vector $\vec{x}$ where the maximum value appears r times, we have:

$$\text{hardmax}_i(\vec{x}) = \begin{cases} \frac{1}{r} & x_i \text{ is the maximum value of } \vec{x} \\ 0 & \text{otherwise} \end{cases}$$

Hahn (2020) builds the strongest argument on the limitations of transformers based on such hard-attention transformers.

Bhattamishra et al. (2020b) proves the Turing-completeness of soft-attention transformers by showing they can simulate RNNs, which are Turing-complete. Specifically, the argument applies to both vanilla transformers with positional encoding, as well as the variant without such encoding and relying only on look-ahead masking for positional information.

Despite the above Turing-completeness results when assuming infinite precision, which is

unrealistic in practical cases, Hahn (2020) found the theoretical limitation of the transformer that prevents it from recognizing periodic regular languages or hierarchichal context-free languages in full generality. Merrill (2019) points out that since the sinusoidal positional encoding is a periodic function, it repeats eventually for long strings, and thus showed that the regular language is not a subset of transformer-recognizable languages, i.e. RL $\not\subseteq$ L(Transformer).

3.2 Recognizing Data and Formal Classes

3.2.1 Regular Languages

Several regular languages have been the subject of studies on sequential models' ability in recognizing them. We look at two subregular categories respectively, namely the star-free languages, and the non-star-free ones.

Star-free languages

Star-free languages are regular languages that involve boolean operations including complementation, concatenation operation, but not the Kleene star operation. For example, $(\mathtt{ab})^*$ is star-free since $(\mathtt{ab})^* = \overline{(\mathtt{b}\overline{\emptyset} \cup \overline{\emptyset}\mathtt{a} \cup \overline{\emptyset}\mathtt{aa}\overline{\emptyset} \cup \overline{\emptyset}\mathtt{bb}\overline{\emptyset})}$, where $\overline{A}$ is the complement of some set A. This is also known as counter-free languages. See Section 3.2.3 for discussions on counters. Jäger and Rogers (2012) argue that constraints in natural language are rarely non-star-free when regular.

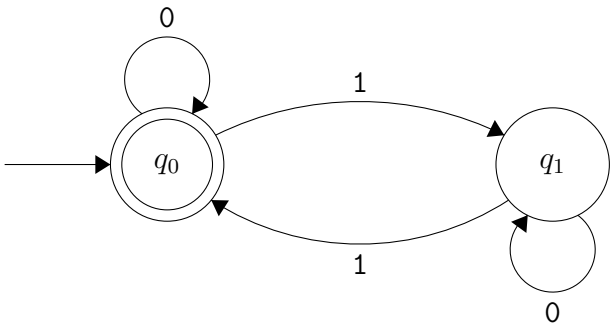
Bhattamishra et al. (2020a) defined[1] languages $\Delta_n = (\mathtt{a}\Delta_{n-1}\mathtt{b})^*$ over $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, and by this definition, n is conveniently the dot depth of the language, where the dot operation is referring to the concatenation operation, since concatenation of $\mathtt{a}$ and $\mathtt{b}$ can be denoted as $\mathtt{a} \circ \mathtt{b}$. Thus, dot-depth measures the nested concatenation, or a hieararchy of star-free languages (Cohen and Brzozowski, 1971). The authors found that the transformer can

[1]The authors denote the language as $\mathcal{D}_n$. However, to avoid confusion with Dyck, greek Delta is used here instead.

recognize $\Delta_1 = (\texttt{ab})^*$, but struggles with Δ_2 onwards, especially when generalizing to longer length, while LSTM can solve the different dot depths perfectly, for both strings with similar length as the training data as well as unseen longer ones.

Non-star-free regular languages

Non-star-free regular languages are also known as periodic regular languages. PARITY is the simplest language in this class. It is the set of bit strings with an even[2] number of 1's, which can be represented by the following FSA with two states:



Recognizing these languages requires modeling periodicity. Hahn (2020) constructs a set of input PARITY strings such that the hard-attention transformer has to misclassify. He fixed a fraction of the strings so that attention is tricked into focusing only on the fraction and the remaining input is ignored, which prevents the model from recognizing PARITY as every bit counts. The inability to compute PARITY also suggests the inability to handle logical formulae as iterated negations require counting whether the number of negations is even or odd to evaluate. On the other hand, self-attention can recognize languages like $\texttt{1}^*$ with ease by focusing on the presence of one symbol that immediately determines non-membership.

Bhattamishra et al. (2020a) studied PARITY and additional non-star-free regular languages such as $(\texttt{aa})^*$ and $(\texttt{abab})^*$ in practical settings, and found that transformers also fail at learning them.

---

[2]Odd should be equivalent here, except that the accepting and rejecting states are flipped.

3.2.2 Dyck and Variants

Following Chomsky and Schützenberger (1963), we define Dyck language as the set of strings that can be reduced to empty by repeatedly deleting a consecutive pair of letters. A more intuitive definition of this is matching k types of parentheses. Dyck-k is a context-free language, and its recognition is of particular interest, since it is a typical hierarchical language. It is interesting to see how a model without recurrence like transformers could handle the language. Dyck data can be generated using a probablistic context-free grammar (PCFG). For example, the PCFG for Dyck-2 with terminals $T = \{\texttt{(}, \texttt{)}, \texttt{[}, \texttt{]}\}$ has the following production rules:

$$S \to (0.25)\ \texttt{(}S\texttt{)}$$

$$S \to (0.25)\ \texttt{[}S\texttt{]}$$

$$S \to (0.25)\ SS$$

$$S \to (0.25)\ \varepsilon$$

Hahn (2020) shows that hard-attention transformer cannot recognize Dyck-2 by constructing a string set that captures the attention, similar to the construction for PARITY above.

Ebrahimi et al. (2020) show that given a starting symbol, a two-layer multi-headed transformer encoder is capable of learning Dyck-k and imperfectly makes generalizations to sequences of longer length. They empirically experimented with Dyck-1 through Dyck-4 and found that transformers outperformed LSTMs except for Dyck-2. The inclusion of a starting symbol is effective for transformers but not LSTMs. The performance of LSTMs also deteriorates as k increases, while transformers demonstrates the opposite.

Hewitt et al. (2020) proposed a variant of Dyck, namely Dyck-(k, m), which is defined as Dyck-k strings with at most m nesting depth. By this definition, Δ_n discussed above is equivalent to Dyck-$(1, n)$. Hewitt et al. (2020) showed that RNNs can generate the language with optimal memory. Yao et al. (2021) studied Dyck-(k, m) on hard-attention transformers.

It was shown that transformers are more efficient in modeling this language than LSTMs. Although both work were motivated by hierarchical constructions in natural language syntax, and some stack-like behavior might be learned, it is worth noting that the bounded hierarchy makes Dyck-(k, m) a regular language.

Papadimitriou and Jurafsky (2020) argue that LSTMs know how to look back at relevant dependencies, thus hierarchy is not as important in such cases for inference even though models may appear to be modeling hierarchical structures. They exprimented with a nested parentheses language, which is equivalent to regular Dyck-$(n, 1)$, as well as shuffle-Dyck, a language with strings like `([)]` that has matching open and closing brackets but not the correct ordering thus has no hierarchical structure.

3.2.3 Counter-acceptable Languages

Following Weiss et al. (2018), a counter is a device that can increment or decrement by a fixed amount, or be compared to 0. A simplified k-counter machine is an FSA that is augmented with k counters. LSTM is a variant of such machine as it designates k dimensions of memory cell $\vec{c}_t$ as counters. Counter languages are defined as languages recognized by a deterministic finite-state automaton with finite unbounded counters. They are supra-regular, i.e., the class is a proper superset of regular languages, while it also spans some context-free and context-sensitive languages. Some simple counter languages include $\mathtt{a}^n\mathtt{b}^n$ (context-free), $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$, $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$ (mildly context-sensitive).

Weiss et al. (2018) studied practical complexity of finite-precision RNNs via their ability to learn $\mathtt{a}^n\mathtt{b}^n$ and $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$. Although LSTM and GRU perform similarly in NLP applications and both utilized gate mechanisms, LSTM is able to implement some counting mechanisms, thus able to recognize the languages, whereas GRUs cannot and are equivalent to finite-state machines.

Suzgun et al. (2019) also studied these simple counter languages on LSTMs, but uses a different sequence prediction task setup that we discuss in the next section, and focused on the effects of the presentation of training data on learnabiltiy, such as the probablistic dis-

tribution of string lengths in the training set. Bhattamishra et al. (2020a) studied additional counter languages using transformers also with the sequence prediction task setup.

3.3 Analysis Methods

The above papers used mainly two different tasks for assessing the language recognition ability, namely the binary classifcation task, and the language transduction task.

3.3.1 Binary Classification Task

Hahn (2020), following Weiss et al. (2018), modeled the task of language recognition as a binary classification task. A model g recognizes a formal language $L \subseteq \Sigma^*$ if $f(g(w)) = 1$ for all and only words $w \in L$, where f is a log-linear classifier or feedforward neural network.

3.3.2 Language Transduction Task

Suzgun et al. (2019); Ebrahimi et al. (2020); Bhattamishra et al. (2020a) modeled language recognition as a transduction task, that is, an auto-regressive model is asked to predict the next symbol given a valid prefix. In case there are multiple valid next symbols, the task is modeled as a multi-label classification problem, where the outputs are k-hot vectors.

For example, when recognizing strings in language $\mathtt{a}^n\mathtt{b}^n$, given the first input character $\mathtt{a}$, the next possible symbol can be either $\mathtt{a}$ or $\mathtt{b}$, but not the end of sequence, thus the expected output is the k-hot vector

$$\begin{bmatrix} \mathtt{a} = 1 \\ \mathtt{b} = 1 \\ [\mathtt{EOS}] = 0 \end{bmatrix} \in \{0,1\}^{|\Sigma \cup \{\mathtt{[EOS]}\}|}$$

whose dimension is the size of the alphabet plus one, where each dimension is for each symbol in the alphabet, with an additional dimension for the `[EOS]` token.

As for the training objective, Suzgun et al. (2019); Bhattamishra et al. (2020a) used the mean-squared error (MSE) between the predicted probabilities and k-hot vectors. Ebrahimi

et al. (2020) used the binary cross entropy (BCE) loss. Bhattamishra et al. (2020a) also tried BCE loss and reported similar performance to the MSE objective.

A noticeable thing for this setup is that this task is learning from positive text only. Gold (1967) has argued that only finite languages can be identified in the limit from positive text only. However, Suzgun et al. (2019) argue that the training scheme is still consistent with Gold's theorem as the training process is giving feedback to the model when it makes errors, thus satisfying the conditions laid out in Angluin (1980) for languages inferrable from positive data.

3.3.3 Probing Internal Representations

In addition to evaluate whether neural sequence models can solve the language recognition task, these works also try to understand what internal mechanisms are used for recognizing these languages.

Weiss et al. (2018) plot the activations of their 10-dimensional LSTM models for $\mathtt{a}^n\mathtt{b}^n$ and $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$, and identified two dimensions that used the counting mechanism, indicating that the LSTM implemented the 2-counter solution for recognizing the languages, while the GRU model visualizations are less intepretable. Similarly, Suzgun et al. (2019) visualized activation values against time steps for their 4-unit LSTM, and discovered that it learns to count up and down upon transition to a different character.

Ebrahimi et al. (2020) visualized the attention maps for each head of their trained transformer model for Dyck language. In their architecture with the starting symbol, the authors have found that the map of one of the heads resembles a stack-based recognizer, where an open bracket attends to itself, and a closing bracket would first pop the stack top, then attends to the next unmatched symbol. However, as the length of the string gets longer, such resemblance also deteriorates.

Bhattamishra et al. (2020a) also studied transformers. The authors visualized the output of the self-attention block, and found that the value for some dimensions match the depth to length ratio for the studied languages. One of the dimensions of the learned positional

encoding for $(\texttt{aa})^*$ also resembles the periodic behavior like $\cos(n\pi)$.

Lastly, although Yao et al. (2021) did not probe their emprically learned model, they proposed constructions that enable transrformer models to learn the bounded Dyck language.

3.4 Summary

Previous work studied computational power of RNNs and transformers using regular, context-free, and counter languages. Although some connections can be drawn between linguistic constructions and these classes, e.g., context-free Dyck resembles hierarchical tree structures in syntax, and regular natural linguistic constraints are usually star-free, these studies are not directly motivated by formal complexity results in natural language syntax or linguistic phenomenon. Our work complements these previous findings by systematically studying simple transformer's ability in recognizing mildly context-sensitive languages, a complexity class that is directly motivated by the hypothesis of natural language complexity such that is approapriate for formal language study because of the costrained generative power, and also significant in theoretical linguistics. We follow task setups in previous work, including the binary classification task and the language transduction task, and look at the transformer's internal representation to understand its mechanism used for recognizing the mildly context-sensitive languages.

Chapter 4

METHODOLOGY

Our experiments seek to answer how transformers perform on learning the mildly context-sensitive languages and making generalizations, as well as what internal mechanisms are used for representing the languages. In this chapter, we describe the two tasks used and their corresponding architectures, and lay out the specific languages studied in the class and methods for generating the training data. Lastly, we discuss how to evaluate the computational power of the trained models.

4.1 Architecture

Although the original transformer is a seq2seq architecture, the model we study is an encoder-only transformer model, similar to how it is used in BERT (Devlin et al., 2019). Unless otherwise noted, the transformer models studied are single-layer, single-head models. We focus on simple models since we are already observing decent performance on them, there is no strong motivation for having more complex models given our focus, but this is nevertheless suitable for future explorations.

We experiment with both binary classification and sequence prediction tasks depending on which is more suitable for each family of languages. Figure 4.1 shows the setup for the binary classication task described in Section 3.3.1. The model classifies whether a string is a member of the taught formal language or not.

The input is the embeddings for tokens in the sequence combined with sinusoidal positional encoding. The embedding layer is initialized using a uniform distribution in the range $[-0.1, 0.1]$. The transformer encoder produces one output embedding for each token. These token embeddings are averaged to create the embedding for the whole sequence. A

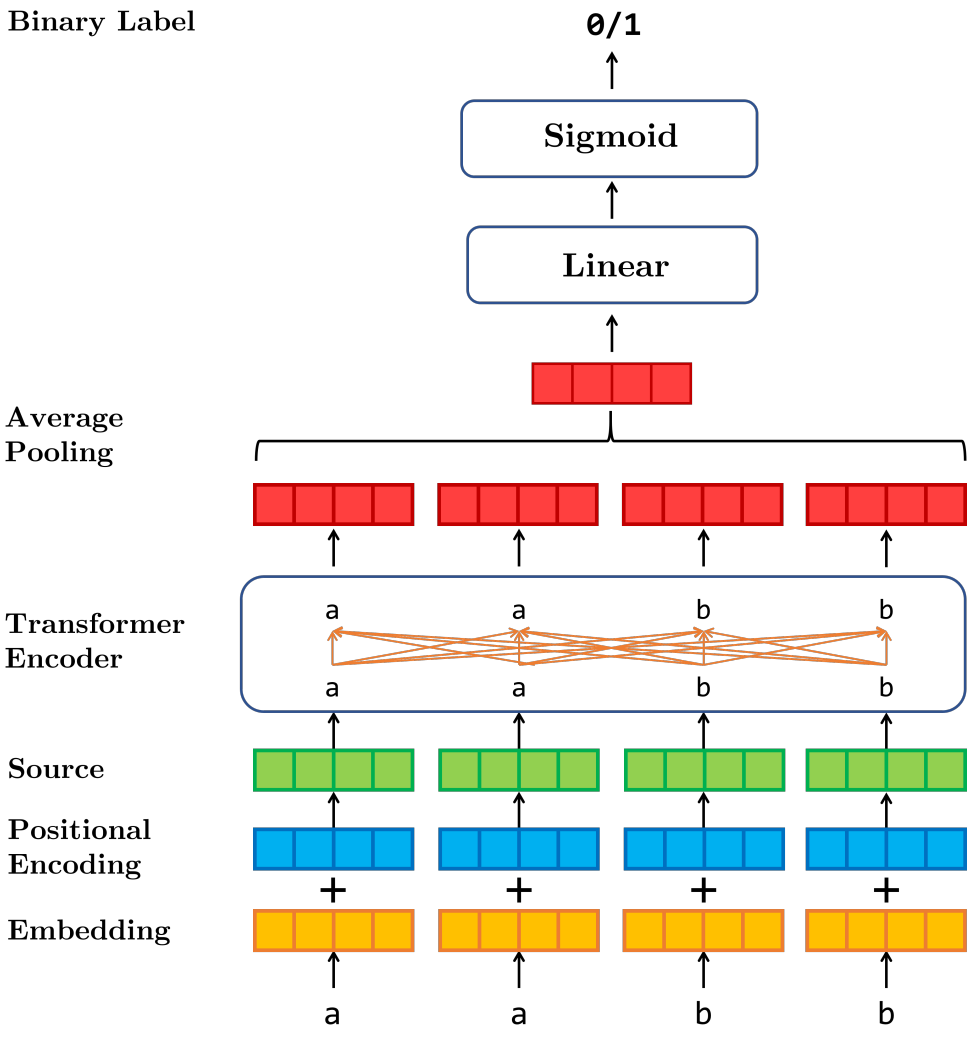


Figure 4.1: Binary classification task setup architecture

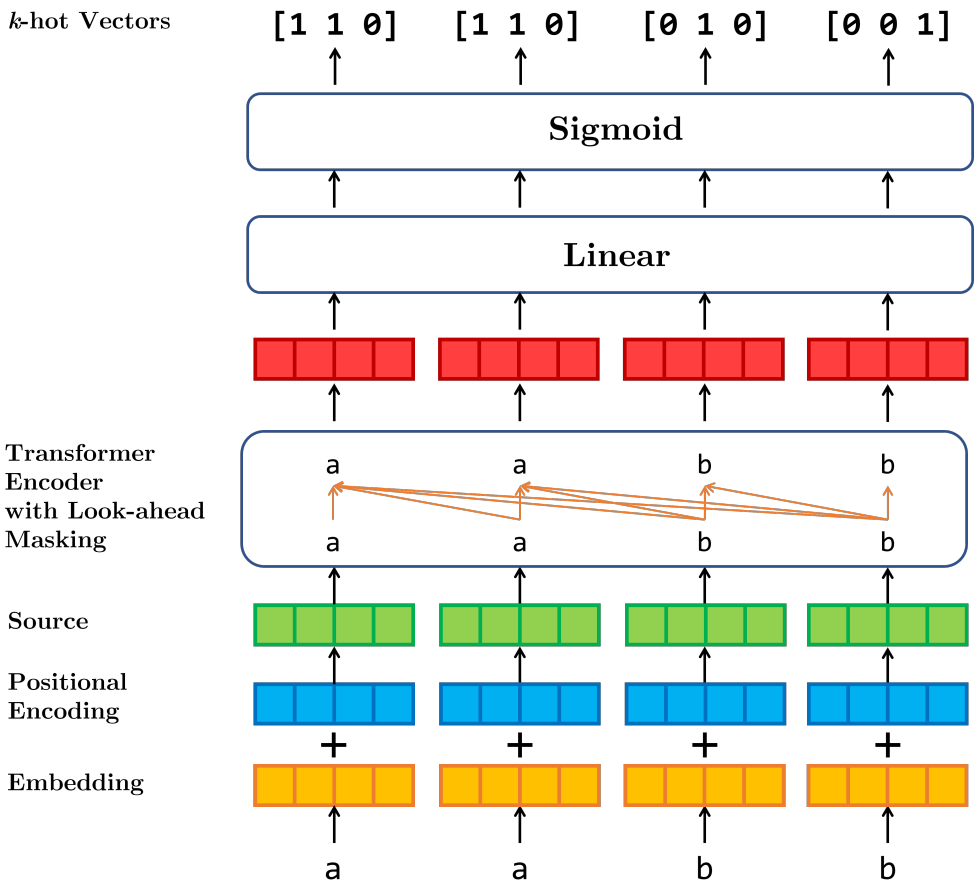


Figure 4.2: Next character prediction task setup architecture

fully connected linear layer maps the pooled embeddings to a single number, which is passed through the sigmoid function to obtain the class labels 0 or 1 with a threshold of 0.5. The loss is the BCE loss between the predicted and target binary class labels.

The second architecture in Figure 4.2 is for the sequence prediction task described in Section 3.3.2. For strings of a language, at each time step, the model learns to predict what the next set of allowed characters is, given the currently seen prefix at the time step. The targets are represented as k-hot vectors, where each vector dimension represents each symbol in the alphabet plus the end-of-sequence token. We explain the scheme for creating the target

k-hot vectors in the following section. It has a few differences from the previous one. Firstly, since the sequence prediction task is like a language modeling setup, we add look-ahead masking to transformer encoder to prevent self-attention attending to later positions. The inclusion of look-ahead masking also provides positional information, thus the sinusoidal positional encoding is optional here. We experiment with cases both with and without the sinusoidal encoding for some of the languages. Then, instead of pooling, all token embeddings are passed through the linear layer and sigmoid function in parallel to obtain k-hot vectors for each time step using a threshold of 0.5. The loss is the individual symbol's BCE loss between the prediction and target k-hot vectors summed.

For each experiment, we also train an LSTM model as a baseline for comparison. The implementations for both transformer and LSTM are the standard ones from PyTorch (Paszke et al., 2019).[1]

4.2 Data

Following the categorization in Genkin et al. (2010), three phenomena and languages in mildly context-sensitive class are of interest here:

1. Copying: ww
2. Cross-serial dependency: $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$
3. Multiple agreements: $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$

All these languages are TAG-recognizable. For each of the three languages, we also compare it with a similar but less complex CFL, as well as a similar but more complex MG-recognizable language, to see if the difference in complexity is correlated with model performance. We describe the data and their generation scheme in more details in the following subsections, and note that all languages are generated exactly once, and used across all hyperparameter tuning and experiment runs. The random seed used for generation is recorded for reproducibility.

[1]The code for the experiments is available at `https://github.com/shunjiewang/mcsl-transformer`

4.2.1 Copying

Copy language $\{ww \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$ is in L(TAG) (Joshi, 1985), and its context-free counterpart is the palindrome language $\{ww^{\mathcal{R}} \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$, where $w^{\mathcal{R}}$ is the reverse of string w. Joshi (1985) indicates that www is not in L(TAG), but w^k for any arbitrary k is in L(MG) (Jäger and Rogers, 2012). Thus, this thesis studies www as the simplest strictly MG-recognizable language for copying.

We use the binary classification task to study this family of languages, since unlike the next two cases we are going to see, there is no simple schema for modeling its recognition as next character prediction. Using binary classification setup allows the self-attention to observe the strings in whole and infer the dependencies among them, since ww can be seen as a cross-serial dependency language with numerous dependencies, that is, each symbol in the first w depends on its copy in the second w and vice versa.

To generate strings in this family, we enumerate each possible w of some length n in range $[1, 11]$, that is, w is a sequence in the Cartesian product $\Sigma^n, \Sigma = \{\mathtt{a}, \mathtt{b}\}$, for all $n \in [1, 11]$. Then, we duplicate the w to yield $ww^{\mathcal{R}}$, ww, and www. The negative examples are random strings sampled from $\{\mathtt{a}, \mathtt{b}\}^*$ with the same lengths as each positive example that are not in the postive examples.

4.2.2 Crossing

Cross-serial language is represented by L(TAG) language $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$. Its context-free counterpart is the nested dependency language $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n$.

For these two languages, we use the language transduction setup, since during the pilot experiments, the trained model using the binary classification setup is likely relying on trivial statistical cues such as consecutive sequences of $\mathtt{b}$'s, instead of truly learning the language for inference. However, we are interested in understanding the underlying mechanism that enables transformers to recognize the languages, rather than just solving our specific task setup and generated data.

In the language transduction task setting, given the prefix of a string, the model has to predict the next set of valid symbols, outputted as a k-hot vector. Following Gers and Schmidhuber (2001), for recognizing the nested language $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n$, when the input is $\mathtt{a}$, the next possible character can be $\mathtt{a}$ or $\mathtt{b}$, and when the input becomes $\mathtt{b}$, the value for n is determined, and the next possible character is now $\mathtt{b}$ or $\mathtt{c}$. Once the input becomes $\mathtt{c}$, the value for m is also determined, and next character is deterministic from this point on. Until we consume the last symbol in the input, we output the end-of-sequence token. Following Suzgun et al. (2019), we denote this scheme as the following:

$$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n \rightarrow (\mathtt{a}/\mathtt{b})^n(\mathtt{b}/\mathtt{c})^m\mathtt{c}^{m-1}\mathtt{d}^n\#$$

where $(\mathtt{a}/\mathtt{b})$ denotes both symbols are in the set of acceptable next characters, and $\#$ denotes the end-of-sequence token. Trivially, we can generalize this scheme to the crossing language $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$:

$$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m \rightarrow (\mathtt{a}/\mathtt{b})^n(\mathtt{b}/\mathtt{c})^m\mathtt{c}^{n-1}\mathtt{d}^m\#$$

4.2.3 Multiple Agreements

The family multiple agreements languages, being simple counter languages, has been extensively studied in Suzgun et al. (2019); Bhattamishra et al. (2020a). We still include this set for completeness, and to complement the previous work, we mainly look at this from the view of Chomsky hierarchy, and also add the L(MG) language in this family, to see how the learning capability might change with added complexity in the formal hierarchy.

Both $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$, $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$ are TAG-recognizable, and their context-free counterpart is $\mathtt{a}^n\mathtt{b}^n$. However, Joshi (1985) suggests $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$ is not in L(TAG), but Stabler (1997) indicates it is in L(MG). Moreover, $a_1^n, ..., a_k^n$ for any arbitrary k is MG-recognizable (Stabler, 1997; Jäger and Rogers, 2012). Thus, this thesis studies $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$ as the simplest strictly MG-recognizable language for multiple agreements family.

Gers and Schmidhuber (2001); Suzgun et al. (2019) has proposed the input-target scheme

for $\mathtt{a}^n\mathtt{b}^n$, $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$, $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$:

$$\mathtt{a}^n\mathtt{b}^n \rightarrow (\mathtt{a}/\mathtt{b})^n\mathtt{b}^{n-1}\#$$

$$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \rightarrow (\mathtt{a}/\mathtt{b})^n\mathtt{b}^{n-1}\mathtt{c}^n\#$$

$$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n \rightarrow (\mathtt{a}/\mathtt{b})^n\mathtt{b}^{n-1}\mathtt{c}^n\mathtt{d}^n\#$$

that is, when the input is symbol a, the next possible characters can be either a or b. As soon as the input becomes b, the value for n is determined, and the recognition becomes deterministic from this point on, until reaching the last symbol in the input sequence which outputs the end-of-sequence token. Trivially, this scheme can be generalized to $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$:

$$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n \rightarrow (\mathtt{a}/\mathtt{b})^n\mathtt{b}^{n-1}\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n\#$$

To summarize, this thesis studies the following languages in Table 4.1.

	CFL	Mildly Context-Sensitive	
	(*less complex*)	L(TAG) (*canonical*)	L(MG) (*more complex*)
Copying	$ww^{\mathcal{R}}$	ww	www (w^k)
Crossing	$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n$	$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$	
Multiple Agreements	$\mathtt{a}^n\mathtt{b}^n$	$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$ $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$	$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$ ($a_1^n...a_k^n$)

Table 4.1: All languages studied in this thesis. Languages in parentheses are for illustrative purpose only.

4.3 Evaluation

To evaluate the trained models' ability in making generalizations, we not only evaluate on the held-out test set that contains sequences with similar lengths as the ones seen during the

training, but also, we evaluate on unseen sequences that are much longer than the ones in the training set. We refer to the former as in-domain evaluation, where the training set and the test set are independently identically drawn from the same distribution, and the latter as out-of-domain evaluation, which is analogous to human learning ability (cf. Linzen (2020))

To understand what mechanisms self-attention has learned to recognize these languages, we also visualize the self-attention maps for each trained transformer model for insights on its learned internal representation like Ebrahimi et al. (2020). We also acknowledge that self-attention pattern is a clue for the internal mechanism but not the whole picture, for example, the feedforward neural network sublayer could also have a role in the language recognition. However, we did observe highly intepretable patterns in just self-attention, thus we focus on this, and leave probing other components for future work.

Chapter 5

EXPERIMENTS

For each of the three families of languages, we first report its performance on in-domain generalization, followed by the visualization of self-attention maps of trained models, and lastly evaluate the models on out-of-domain data.

Hyperparameters are tuned for each of the languages studied and the set that ends up with the smallest development loss is selected. The optimizer is the AdamW optimizer (Loshchilov and Hutter, 2019). Experiments are run three times with different random seeds and the average of the results is reported.

5.1 Copying

The copying languages are trained using the binary classification task setup. The generated data has $|w|$ in range $[1, 11]$. That gives us 4089 positive examples, and we match them with equally many negative examples. We then do a 70%-15%-15% train-dev-test random split of the dataset.

We tune the learning rate selected from $\{1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}\}$ and d_{model} from $\{16, 32, 64\}$. Training uses early stopping with a patience of 10. The test accuracies averaged across three runs with different random seeds are shown in Table 5.1.

The results show that transformer performed decently on the test set and is comparable to LSTM, and interestingly, the less complex context-free language seems harder to learn than the more complex ones. For this set of experiements, the variance of test accuracy among the three runs is low, which is very different from the other two families of languages trained using the language transduction task described in the next sections.

To understand what strategy is used for representing the language, we visualize the self-

	$ww^{\mathcal{R}}$	ww	www
Transformer	93.4 (max=93.7, σ=0.6)	97.2 (max=97.3, σ=0.1)	97.6 (max=97.9, σ=0.3)
LSTM	95.3 (max=97.6, σ=1.7)	98.6 (max=98.6, σ=0.04)	98.8 (max=99.0, σ=0.2)

Table 5.1: Copying languages in-domain test accuracy (%)

attention maps for the three transformer models. Recall the x-axis is the key and y-axis is the query. Each row is a distribution of attention weights from one symbol to itself and every other symbol, which sums to 1. We use $w =$ `aabaaabbaaa` as an example in all visualizations.

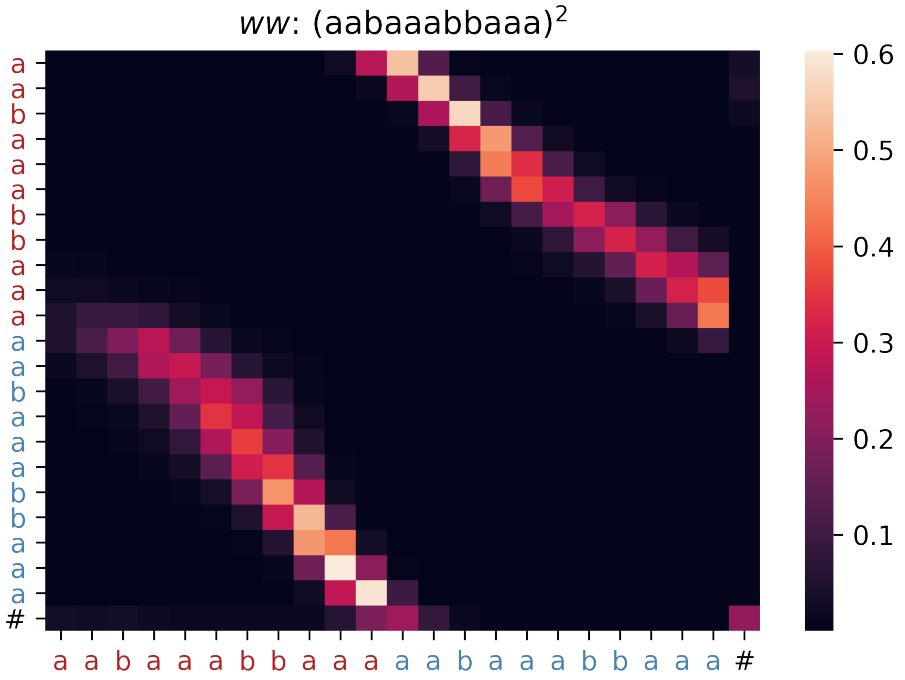


Figure 5.1: ww self-attention map

Figure 5.1 suggests the self-attention has cleared learned a strategy to represent reduplicated patterns in ww strings. It has identified the midpoint of the string, and each symbol in one w attends to its copy in the other w, resulting in this double diagonal pattern.

For $ww^{\mathcal{R}}$ in Figure 5.2, we also get a similar pattern as ww, although less neat, once again suggesting the context-free language is harder to learn than the more complex ones. Only one of the two diagonals is clear, but both are pointing to the correct anti-diagonal direction, suggesting the symbols are correctly attending to their corresponding symbols in the mirrored string.

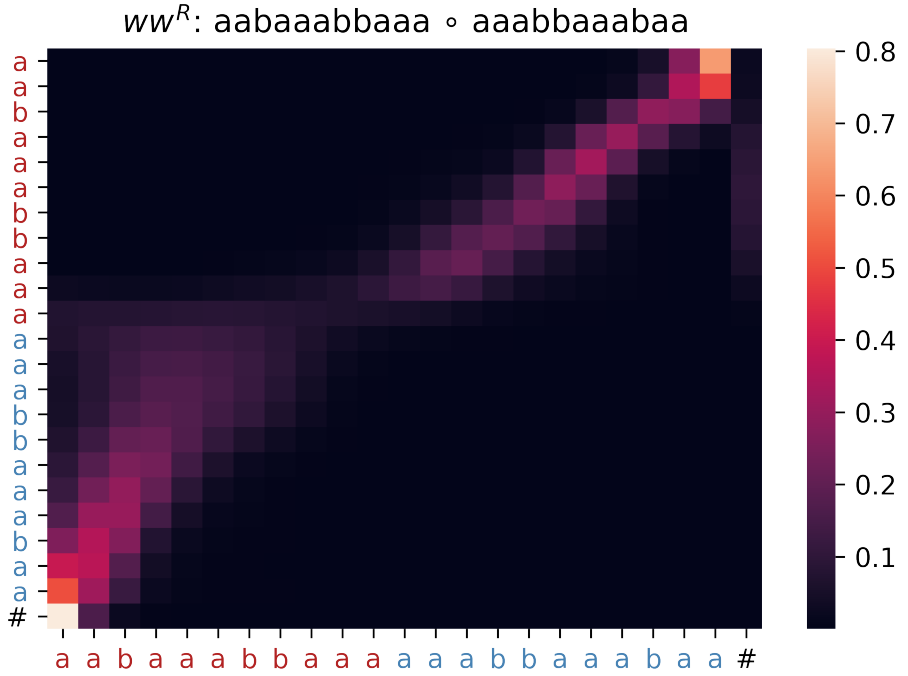
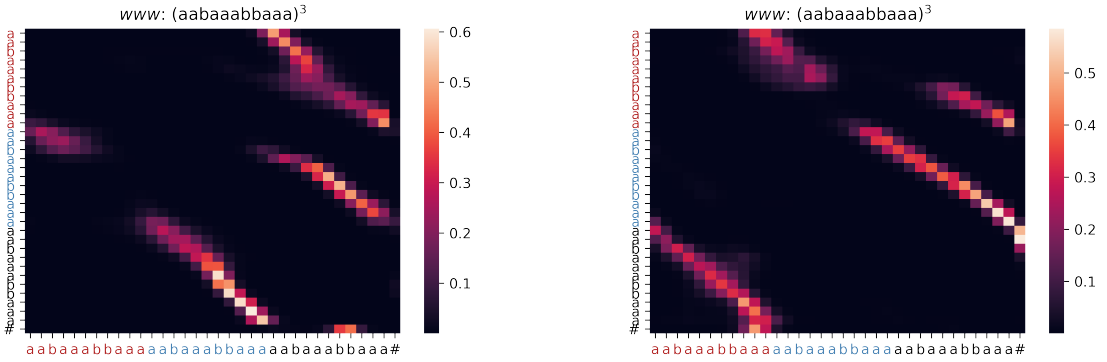


Figure 5.2: $ww^{\mathcal{R}}$ self-attention map

Interestingly, $ww^{\mathcal{R}}$ being a typical center-embedded context-free language, does not demonstrate any behavior of simulating a stack when trained using our simple transformer, but has a self-attention pattern that is very similar to the more complex ww. This is different



(a) Run 1

(b) Run 2

Figure 5.3: www self-attention maps with different random seeds

from theoretical and empirical studies of Dyck languages such as Ebrahimi et al. (2020); Yao et al. (2021) which rely on a stack-based recognizer in one of the heads for recognizing such context-free languages.

Lastly, we visualize the models for www in Figure 5.3. We observed two patterns in two runs with diffrent random seeds and present both of them. In either map, we see three clearest diagonals, and some partial diagonals, suggesting each w is at least attending to one of the other w's. Boundaries between each w were clearly identified.

Since self-attention is not necessarily symmetrical, in an otherwise perfect setting, one would expect each w to attend to the other two, resulting in six diagonals in total. However, this map suggests the self-attention might learn only three such dependencies. We do consistently get at least 3 clearer diagonals, which should be the minimum number required for making reliable inferences for www.

To summarize the above maps, self-attention seemed suitable for representing copying patterns, and generalizes decently to unseen in-domain data. To see if this learned representation generalizes to strings of higher length, we evaluate the model on additional out-of-domain data where $|w|$ is 12 and 13. All permutations with these lengths were generated as

		$\|w\|$	
		12	13
$ww^{\mathcal{R}}$	Transformer	50.5 (max=51.0, σ=0.4)	50.0 (max=50.6, σ=0.4)
	LSTM	95.7 (max=98.1, σ=1.7)	95.6 (max=98.1, σ=1.8)
ww	Transformer	50.2 (max=50.3, σ=0.1)	50.2 (max=50.3, σ=0.1)
	LSTM	95.2 (max=96.2, σ=0.7)	84.3 (max=88.2, σ=2.8)
www	Transformer	50.5 (max=51.1, σ=0.5)	50.8 (max=50.9, σ=0.5)
	LSTM	96.9 (max=98.4, σ=1.2)	82.6 (max=85.9, σ=2.4)

Table 5.2: Copying languages out-of-domain test accuracy (%) (null accuracy = 50%)

positive examples. The result is in Table 5.2. Notice the null accuracy is 0.5 for this binary classification task.

Despite the learned self-attention pattern is largely explainable, and the in-domain test accuracy is satisfying, the transformer models cannot generalize to longer out-of-domain sequences that they were never trained on, and the inference has immediately become a random guess even for w strings that are only 1 or 2 bits longer than the ones seen in the training set. Meanwhile, LSTM still demonstrates good generalization behavior on these out-of-domain ones.

5.2 Crossing

The crossing language $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$ is trained with its context-free counterpart $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n$ using the language transduction setup. The generated datasets have n, m in range $[2, 50]$, totaling 2401 positive examples for each language. Recall from Section 3.3.2 that the language transduction task uses positive examples only. We also do a 70%-15%-15% train-dev-test random split of the dataset.

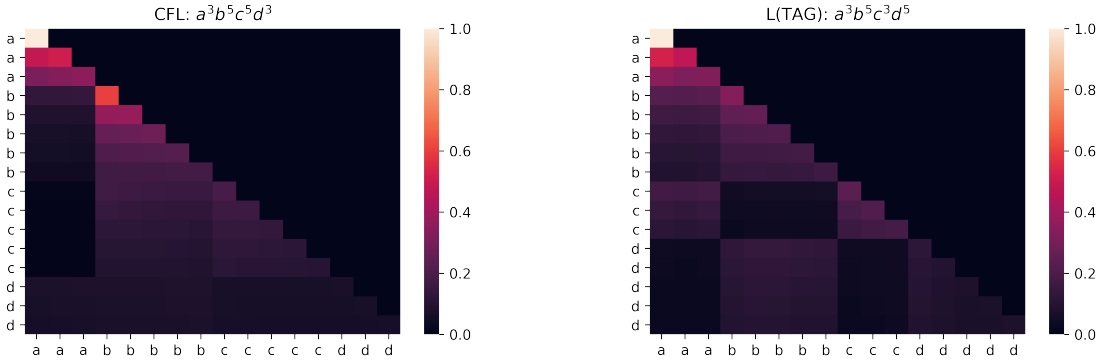
We experiment with both configurations with and without a sinusoidal positional encoder and see how the performance differs in the two cases. The case without sinusoidal encoding will only take advantage of positional information from the look-ahead masking. The training uses early stopping with a minimum delta of 1×10^{-4}, and a patience of 20, to stop early once the learning curve has already become very flat and the change is very subtle. We tuned learning rate in $\{1 \times 10^{-2}, 1 \times 10^{-3}\}$, and d_{model} in $\{16, 32, 64\}$. Again, three runs with different random seeds and otherwise identical hyperparameters were performed, and we report the following average test accuracy in Table 5.3:

	$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n$	$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$
Transformer w/ PE	100.0 (max=100.0, σ=0.0)	99.4 (max=99.7, σ=0.3)
Transformer w/o PE	100.0 (max=100.0, σ=0.0)	100.0 (max=100.0, σ=0.0)
LSTM	99.9 (max=100.0, σ=0.1)	100.0 (max=100.0, σ=0.0)

Table 5.3: Nesting and crossing languages in-domain test accuracy (%)

For the two languages, both transformers with or without positional encoding generalize well on in-domain test set, and is comparable to LSTM. The variance of test accuracy is also low across the three runs. We once again visualize the self-attention maps of the four

models. Notice that with the look-ahead masking, only the lower triangular half will have values. We use $n = 3, m = 5$ as an example for all visualizations. Figure 5.4 is for the transformer model with sinusoidal positional encoding:



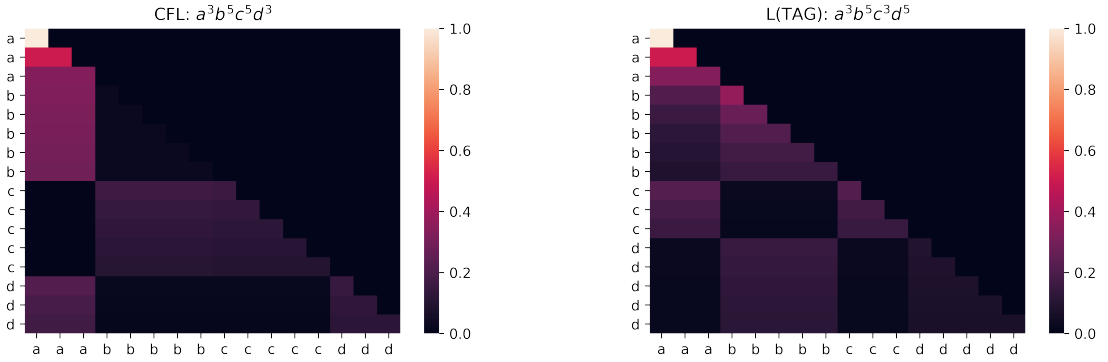
(a) Nesting with PE

(b) Crossing with PE

Figure 5.4: Self-attention maps for nesting and crossing languages with sinusoidal positional encoding

The self-attention map in subfigure (b) for crossing language has a checkboard-like pattern, since each subsequence of symbols has higher attention on their dependents, and less so on others, as evident by the bottom rows of `d`'s. For the context-free nesting language in subfigure (a), rows of dependents `b`'s and `c`'s behave in concordance by having similar attention weights on their non-dependents `a`.

As for the transformer without positional encoding in Figure 5.5, we observe some more explainable patterns:



(a) Nesting without PE

(b) Crossing without PE

Figure 5.5: Self-attention maps for nesting and crossing languages without sinusoidal positional encoding

The first major difference from previous cases is that the omission of sinusoidal encoding enabled each symbol to have uniform attention within each subsequence. Thus, instead of pixel grids in the previous case, we observe segments of horizontal bars. In addition, we can observe that the strategies used for recognizing nesting and crossing strings are very similar, although the two languages already belong to two different complexity classes.

We then evaluate these models' performance on out-of-domain data. We take strings with n, m in range $[51, 100]$ and $[101, 150]$ respectively. Some experiement runs demonstrate high variance in test accuracies, even though the runs only differ in random seeds and have otherwise identical hyperparameters. Thus, we would like to highlight the maximum accuracy and the standard deviation of the test accuracy for the three runs with different seeds.

From Table 5.4, we see that the omission of sinusoidal positional encoding in transformers and relying only on positional information from look-ahead masking have significant improvements on generalization to longer sequences, although the accuracies are still not as good as the best LSTM models. This improvement is consistent with the findings in

		n, m range	
		$[51, 100]$	$[101, 150]$
$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n$	Transformer w/ PE	10.2 (max=14.1, σ=2.8)	0.0 (max=0.0, σ=0.0)
	Transformer w/o PE	91.6 (max=94.1, σ=2.4)	1.3 (max=2.6, σ=1.1)
	LSTM	79.7 (max=100.0, σ=28.6)	66.7 (max=100.0, σ=47.1)
$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$	Transformer w/ PE	0.0 (max=0.0, σ=0.0)	0.0 (max=0.0, σ=0.0)
	Transformer w/o PE	94.4 (max=100.0, σ=5.9)	10.6 (max=31.9, σ=15.0)
	LSTM	89.3 (max=100.0, σ=15.2)	21.5 (max=50.7, σ=21.4)

Table 5.4: Nesting and crossing languages out-of-domain test accuracy (%). Notice the improvement by removing positional encoding. Transformer is still not as good as the best LSTM models.

Bhattamishra et al. (2020a) for a number of languages.

5.3 Multiple Agreements

The multiple agreements languages are also trained using the language transduction setup. The training for this set of languages is particularly hard since the dataset is very small: only 1 example is available for each n. Unlike Weiss et al. (2018); Suzgun et al. (2019); Bhattamishra et al. (2020a), which duplicate the training examples subject to some probablistic distribution, we are only using each example once in the generated dataset. The generated dataset has n in range $[2, 50]$. Thus, the generated dataset has 49 positive examples in total. Therefore, we specifically chose sequences with $n \in \{5, 15, 25, 35, 45\}$ as the development set, and sequences with $n \in \{6, 16, 26, 36, 46\}$ as the test set, and the remaining 39 examples as the training set. This is to prevent sets obtained from random split containing samples biasing towards longer or shorter sequences on one end.

For better convergence given such a small dataset, this family of languages is trained using early stopping with a minimum delta of 1×10^{-4}, and a more generous patience of 400, since each training epoch contains only 39 samples, it requires more epochs to train than the previous two families.

In addition to the simple single-layer, single-head transformer, we also experiment with a two-layer transformer with four heads per layer, to see if the generalization is improved with added layers and heads.

During pilot experiments, we found that convergence with the positional encoding is hard. Thus, for this set of experiments, we study the transformer without sinusoidal positional encoding and only rely on positonal information from look-ahead masking.

We tuned learning rate in $\{1 \times 10^{-2}, 1 \times 10^{-3}\}$, and d_{model} in $\{16, 32, 64\}$. The following test accuracy on in-domain test set in Table 5.5 is obtained, averaged from three runs with different random seeds.

All models generalized perfectly to the small yet unseen test set. We note that this testing setup is also different from Suzgun et al. (2019); Bhattamishra et al. (2020a), since their

	$\mathtt{a}^n\mathtt{b}^n$	$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$	$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$	$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$
Transformer 1-1	100.0	100.0	100.0	100.0
Transformer 2-4	100.0	100.0	100.0	100.0
LSTM	100.0	100.0	100.0	100.0

Table 5.5: Multiple agreements languages in-domain test accuracy (%)

experiments have no test set held out from the in-domain dataset and all examples were seen during training, so the evaluation was mainly on its generalization to out-of-domain data.

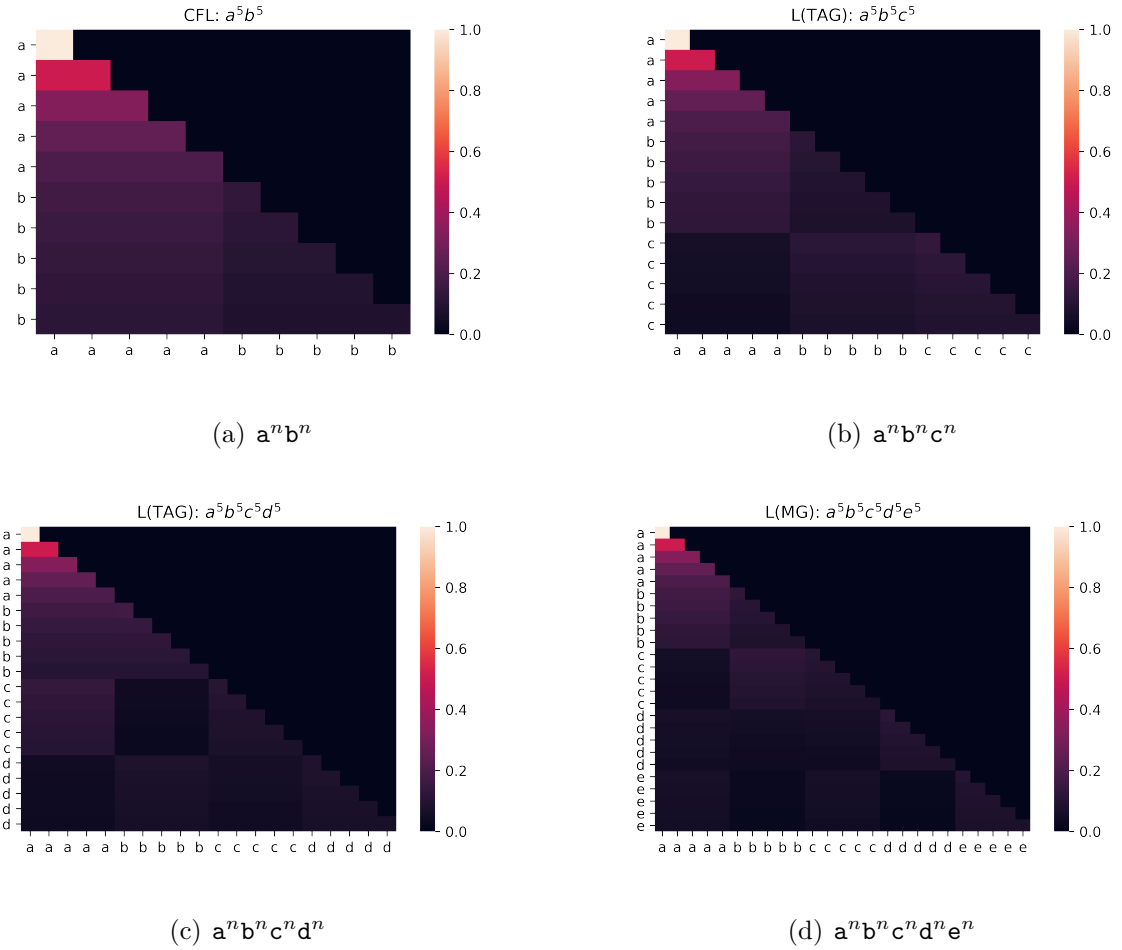
We visualize the self-attention maps of simple transformer models with $n = 5$ to see the strategy it is using in Figure 5.6, again notice only the lower triangular half has values because of the look-ahead masking.

We can observe that the four models are using similar strategies, despite these four languages are spanning three different complexity classes on our extended formal hierarchy. Each symbol attends to each subsequnce of other symbols using a different weight. Within each subsequence, the attention is uniform, similar to the above case for crossing language without positional encoding. Thus, we also get horizontal bars and square blocks formed by these bars.

Since the in-domain test set is very small, we then turn to out-of-domain evaluation. The results are in Table 5.6.

Both transformer and LSTM models have high variance in test accuracy for this set of experiments even though the models only differ by random seeds. When n is in range $[51, 100]$, the best transformer models all performed relatively well and are comparable to the best LSTM models, suggesting transformer's learned representation can be generalized to some longer sequences.

However, for n in range $[101, 150]$, the performance of transformer models deteriorates with added complexity. It was only able to give decent performance on the less complex



(a) $\mathtt{a}^n\mathtt{b}^n$

(b) $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$

(c) $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$

(d) $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$

Figure 5.6: Self-attention maps for multiple agreements languages without sinusoidal positional encoding

		n range	
		$[51, 100]$	$[101, 150]$
$\mathtt{a}^n\mathtt{b}^n$	Transformer 1-1	100.0 (max=100.0, σ=0.0)	93.2 (max=100.0, σ=9.6)
	Transformer 2-4	98.0 (max=100.0, σ=2.9)	65.3 (max=100.0, σ=46.2)
	LSTM	100.0 (max=100.0, σ=0.0)	100.0 (max=100.0, σ=0.0)
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$	Transformer 1-1	93.2 (max=100.0, σ=9.6)	17.7 (max=49.0, σ=22.2)
	Transformer 2-4	100.0 (max=100.0, σ=0.0)	14.3 (max=28.6, σ=11.7)
	LSTM	100.0 (max=100.0, σ=0.0)	14.3 (max=28.6, σ=11.7)
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$	Transformer 1-1	71.4 (max=81.6, σ=13.0)	0.0 (max=0.0, σ=0.0)
	Transformer 2-4	75.5 (max=100.0, σ=17.6)	2.7 (max=8.2, σ=3.8)
	LSTM	100.0 (max=100.0, σ=0.0)	38.1 (max=67.3, σ=23.3)
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$	Transformer 1-1	75.5 (max=100.0, σ=18.0)	10.9 (max=32.7, σ=15.4)
	Transformer 2-4	92.5 (max=95.9, σ=3.5)	0.0 (max=0.0, σ=0.0)
	LSTM	66.7 (max=100.0, σ=27.5)	33.3 (max=100.0, σ=47.1)

Table 5.6: Multiple agreements languages out-of-domain test accuracy (%)

context-free $\mathtt{a}^n\mathtt{b}^n$, but in general struggles with the L(TAG) and L(MG) ones. The multi-headed model is not better than the simple model in this range, which is somewhat surprising. Slightly better performance can only be observed for $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$, and overall the performance is worse than the simple model in the other three cases. However, the poor performance of the multi-headed model can be due to a lot of factors, for example, our training dataset is very small, thus some additional comparisons between the simple and multi-headed models are suitable for future work.

Chapter 6

DISCUSSION

6.1 Strategy: Representation and Power of Transformers

In general, we observe good performance of simple transformer models on all mildly context-sensitive languages studied, for both L(TAG) and L(MG). Self-attention was able to develop highly explainable representations for these languages. The performance is also comparable to LSTMs, while LSTMs performed more robustly on generalization to out-of-domain data.

The learned representation for reduplication clearly took advantage of the attention mechanism, which was initially designed to align source and target sentences in machine translation. Self-attention was able to handle multiple alignments in either main digonal and anti-diagonal directions in the reduplicated patterns. However, the added complexity by L(MG) indeed posed additional challenges in representing them, as evident by the non-optimal representation learnt for www.

For the reduced complexity case, no significant differences in complexity of the representation are observed between CFL and L(TAG), despite the former being less complex on the formal hieararchy, nor is any stack-like behavior observed for solving the CFLs using the simple transformers. The findings in Ebrahimi et al. (2020); Yao et al. (2021) suggest that a stack-like behavior is likely to occur in the second layer of a two-layer transformer as it produces the hard attention. It would be an interesting question to ask then whether a stack-like representation, or the representation we observed empirically is optimal for representing these CFLs. Our learned representations may or may not be the optimal mechanism, and there are certainly equally good or simpler methods available. For example, Hahn (2020) suggest $\mathtt{a}^n\mathtt{b}^n$ is easy to recognize by fixing on a few inputs to force reject the language, and Merrill (2021) suggested a counting strategy that allows transformers to recognize $\mathtt{a}^n\mathtt{b}^n$.

However, whether learning such strategies is easy or not is another thing, and what specific architecutral design is required to enable them, for example, special positional encoding instead of vanilla sinusoidal encoding, specific number of heads and layers, is a separate task to tackle. Yao et al. (2021) also suggest such hard attention pattern is hard to learn for Dyck as input depth increases.

The comparison of crossing languages with optional sinusoidal positional encoding suggested a limitation of the vanilla positional encoding. We observed better performance without them for generalizing to longer sequences. The absence of recurrence in transformer models required positional information to be incorporated in some other way, but the vanilla sinusoidal one has its inefficiencies, e.g., for being periodic, and in our case, does not aid generalization. Bhattamishra et al. (2020a) suggest this is due to the longer sequences with such positional encoding were not seen during training time, and the model without the encoding is less sensitive to such problem thus performing better. However, although look-ahead masking worked well in our test, it is only appropriate in auto-regressive language modeling settings, and cannot be taken advantage of by models like BERT.

Lastly, we tested the multi-head setup for multiple agreements languages, however, the usefulness of multi-headedness is not immediately evident for this set of languages. We did not push too hard on comparing single-headed versus multi-headed since simple models already performed well enough and was informative for our study. However, a more structured comparison can be done in the future to see if this is a language-specific situation.

6.2 Linguistic Significance

We now take a step back to look at the broader picture and think about how powerful a model needs to be for modeling human language.

If the length of natural language strings are bound by a very large number, then natural language can be processed by a model as weak as a finite-state automaton, although such finite automata will be very large in this case (Mohri and Sproat, 2006). This is also why finite state processing of natural language is still promising in some use cases. Corpora studies

have found some boundedness in syntax as well, for example, the depth of multiple center-embedded clauses is exactly 3 in written language, and is technically non-existent in spoken language, thus reducing the contex-free pattern to a regular one (Karlsson, 2007). Similarly, Shieber (1985) indicates judgments beyond triply embedded clauses by native Swiss German speakers get weaker, and effectively reduce the context-sensitive language to context-free if such a finite bound indeed exist. Thus, if a model is only trained and applied on naturally occuring sentences, it may not need to be responsible for recursion or other infinities (cf. Branco 2018, Chapter 5). So, if transformers can model strings up to a finite bound, it could still have success in modeling human language despite the theoretical limitations in self-attention, and the disparity between theory and practical success no longer constitute a problem. For example, our experiments indicate that the vanilla transformer generalizes well on ww up to the length it has seen during training, and if we never touch on longer sequences, we are not concerned with its failure on out-of-domain data.

This is what motivated the study of bounded-depth Dyck languages in Hewitt et al. (2020); Yao et al. (2021). However, such a bounding constraint indeed weakens the argument about the model's ability to process hierarchy since the bounded Dyck is a regular language and a regular, non-hierarchical analysis can be given to any member of the language. This is similar to how the Dutch argument on cross-serial dependencies can be challenged by an alternative, less complex analysis. Unboundedness may also have some usefulness in practical NLP applications, for example, Weiss et al. (2018) suggest processing linearized parse trees requires unbounded counting of brackets and nesting depths. Also, Shieber (1985) suggests going down the bounded path or arguing for it is unreasonable as it enables the proof of natural language to be regular or even finite. This effectively demeans any syntactic theory. In general, it is still worthwhile for modern NLP models to have robust behaviours and generalization to infinities just like the grammar formalisms are, instead of only modeling finite patterns.

This thesis compares the power of transformers against that of MCSG formalisms, which is motivared by a hypothesis for natural language complexity, and is linguistically interesting

as well as computationally efficient. However, these formalisms could also have generative power beyond naturally occuring data. For example, the palindrome language $ww^{\mathcal{R}}$ is unattested in naturally occuring sentences, unless in a deliberately constructed one, although it is well licensed by CFG, TAG, and MG. Thus, accepting an MCS account of natural language will have the byproduct of licensing such unattested languages (cf. Kobele (2006)). Therefore, it is still a huge and difficult open question that what should the relationships be among the powers of NLP models, grammar formalisms, natural language, and naturally occuring data.

6.3 Limitations of Analysis Method

A deficiency of learning from finite examples is that there is always some maximum sentence length in our training data, thus could potentially weaken our argument if we are bound by it, as discussed in the previous subsection. Although we can always look at the out-of-domain generalization, we still need theoretical support on how experiments based on finite examples can draw conclusions without loss of generality.

There are aspects to improve in our experiment setup. The test accuracies for our out-of-domain experiments have very high variance. Better training processes can be explored for more stable training and more consistent results. The three families of languages are not studied using the same setup. The transduction setup, although having promising test results, is hard for a targeted evaluation to be performed for verifying if the model is relying on heuristics for inference. We hope to find a unified task for them for enabling better comparisons.

There are other components to look at in addition to the attention maps, as we acknowledged in Section 4.3, for example, Hahn (2020); Yao et al. (2021) proposed constructions that make use of feedforward neural network sublayer for language recognition in additonal to the self-attention. Future studies of stuctural probes can be performed to understand the use of these components. Also, we only focused on the simple vanilla transformer in this work, but a multi-layer multi-head transformer may demonstrate a different mechanism

for representing these languages. Such a different architecuture setup can also be tested to better understand the usefulness of multi-head attention.

Chapter 7

CONCLUSION

This thesis studies the expressive power of transformer models for NLP tasks by comparing them against mildly context-senstive grammar formalisms, which are hypothesized to be capturing the complexity of natural language. Specifically, we studied three TAG-recognizable languages, namely copying, crossed dependency, and multiple agreements. For each language, we also compare it with a less complex context-free language and a more complex MG-recognizable language to assess the relationships between expressiveness and varying levels of complexity.

We experimented with both binary classification and language transduction task setups depending on which is more suitable for each family of languages. In general, simple vanilla transformer models already performed well on recognizing these languages when tested on in-domain generalization, and are comparable to LSTMs. The self-attention patterns learned are highly explainable for the trained models. However, their ability to make generalizations to longer sequences can be limited and are often outperformed by LSTMs. This could be due to the limitation of the default sinusoidal positional encoding.

There are still many interesting dimensions and directions to explore on this topic. This work focused on learning languages from finite examples empirically, a natural extension is to give formal proofs or constructions on how transformers can learn MCSL in general, if theoretically permitting, like Yao et al. (2021) did for Dyck-(k, m). Another natural extension is to experiment with varied architectural configurations. This work only studied simple vanilla transformers since they are already performing well on the held-out test set, but it is still worth exploring what effects different positional encoding and other components, or varied model hyperparameters have on the performance.

We only experiemented with the simplest languages in $L(\text{MG})$, and more explorations can be done in this class for evaluating the model's ability in handling increasingly complex patterns. The crossing and multiple agreements languages were trained using the transduction setup, and how to teach these languages using both positive and negative examples while preventing the models from resorting to heuristics could also be looked at. Once we know how to train these two families using the binary classification task setup, two order-invariant languages can be included in the study. The shuffle of $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$, namely $O_2 = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}^* \mid |w|_a = |w|_c \wedge |w|_b = |w|_d\}$, as well as the shuffle of $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$, namely $\text{MIX} = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid |w|_a = |w|_b = |w|_c\}$ are both 2-MCFL (Salvati, 2015), thus are mildly context-sensitive. They are useful for targeted evaluation in assessing the relationships between the canonical MCSL and its shuffled, order-invariant form, to see whether the model has, e.g., inductive bias in counting, and at the same time, being MCSLs themselves make them consistent with the class of languages that is of interest in this work.

BIBLIOGRAPHY

Angluin, D. (1980). Inductive inference of formal languages from positive data. *Information and Control*, 45(2):117 – 135.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. arXiv: `1607.06450 [stat.ML]`.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Becker, T., Joshi, A. K., and Rambow, O. (1991). Long-distance scrambling and Tree Adjoining Grammars. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics*, Berlin, Germany. Association for Computational Linguistics.

Bhatt, R. and Joshi, A. (2004). Semilinearity is a syntactic invariant: A reply to Michaelis and Kracht 1997. *Linguistic Inquiry*, 35(4):683–692.

Bhattamishra, S., Ahuja, K., and Goyal, N. (2020a). On the ability and limitations of transformers to recognize formal languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Online. Association for Computational Linguistics.

Bhattamishra, S., Patel, A., and Goyal, N. (2020b). On the computational power of transformers and its implications in sequence modeling. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 455–475, Online. Association for Computational Linguistics.

Branco, A. (2018). Computational complexity of natural languages: A reasoned overview. In *Proceedings of the Workshop on Linguistic Complexity and Natural Language Processing*, pages 10–19, Santa Fe, New-Mexico. Association for Computational Linguistics.

Bresnan, J., Kaplan, R. M., Peters, S., and Zaenen, A. (1982). Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13(4):613–635.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Chomsky, N. (1956). Three models for the description of language. *IEEE Transactions on Information Theory*, 2(3):113–124.

Chomsky, N. (1957). *Syntactic Structures*. Mouton, The Hague.

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

Chomsky, N. (1963). Formal properties of grammars. In R. Duncan Luce, R. B. and Galanter, E., editors, *Handbook of Mathematical Psychology*, volume 2, page 323–418. Wiley and Sons, New York.

Chomsky, N. and Schützenberger, M. (1963). The algebraic theory of context-free languages. In Braffort, P. and Hirschberg, D., editors, *Computer Programming and Formal Systems*, volume 35 of *Studies in Logic and the Foundations of Mathematics*, pages 118–161. Elsevier.

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

Cohen, R. S. and Brzozowski, J. (1971). Dot-depth of star-free events. *Journal of Computer and System Sciences*, 5(1):1–16.

Culy, C. (1985). The complexity of the vocabulary of Bambara. *Linguistics and Philosophy*, 8(3):345–351.

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

Ebrahimi, J., Gelda, D., and Zhang, W. (2020). How can self-attention networks recognize Dyck-n languages? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4301–4306, Online. Association for Computational Linguistics.

Genkin, D., Francez, N., and Kaminski, M. (2010). *Mildly Context-Sensitive Languages via Buffer Augmented Pregroup Grammars*, pages 144–166. Springer Berlin Heidelberg, Berlin, Heidelberg.

Gers, F. A. and Schmidhuber, E. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. *Trans. Neur. Netw.*, 12(6):1333–1340.

Gold, E. M. (1967). Language identification in the limit. *Information and Control*, 10(5):447 – 474.

Graf, T. (2021). Minimalism and computational linguistics. lingbuzz/005855.

Hahn, M. (2020). Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Harkema, H. (2001). A characterization of minimalist languages. In de Groote, P., Morrill,

G., and Retoré, C., editors, *Logical Aspects of Computational Linguistics*, pages 193–211, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hartmanis, J. and Shank, H. (1968). On the recognition of primes by automata. *J. ACM*, 15(3):382–389.

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

Hewitt, J., Hahn, M., Ganguli, S., Liang, P., and Manning, C. D. (2020). RNNs can generate bounded hierarchical languages with optimal memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1978–2010, Online. Association for Computational Linguistics.

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

Hunter, T. (2021). The Chomsky hierarchy. In Allott, N., Lohndal, T., and Rey, G., editors, *A companion to Chomsky*, Blackwell companions to philosophy. Wiley Blackwell, Hoboken, NJ.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France. PMLR.

Joshi, A. K. (1985). *Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions?*, page 206–250. Studies in Natural Language Processing. Cambridge University Press.

Joshi, A. K. (1987). Unification and some new grammatical formalisms. In *Theoretical Issues in Natural Language Processing 3*.

Jäger, G. and Rogers, J. (2012). Formal language theory: refining the Chomsky hierarchy. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1598):1956–1970.

Kallmeyer, L. (2010). *Parsing Beyond Context-Free Grammars*. Springer Publishing Company, Incorporated, 1st edition.

Karlsson, F. (2007). Constraints on multiple center-embedding of clauses. *Journal of Linguistics*, 43(2):365–392.

Kobele, G. M. (2006). *Generating copies: An investigation into structural identity in language and grammar*. PhD thesis, University of California, Los Angeles.

Lin, T., Wang, Y., Liu, X., and Qiu, X. (2021). A survey of transformers. arXiv: `2106.04554 [cs.LG]`.

Linzen, T. (2020). How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, Online. Association for Computational Linguistics.

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

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Manning, C. D., Clark, K., Hewitt, J., Khandelwal, U., and Levy, O. (2020). Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054.

Merrill, W. (2019). *Sequential neural networks as automata*. Bachelor's Thesis, Yale University.

Merrill, W. (2021). Formal language theory meets modern NLP. arXiv: `2102.10094 [cs.CL]`.

Michaelis, J. (1998). Derivational minimalism is mildly context–sensitive. In Moortgat, M., editor, *Logical Aspects of Computational Linguistics*, pages 179–198, Berlin, Heidelberg. Springer Berlin Heidelberg.

Michaelis, J. and Kracht, M. (1997). Semilinearity as a syntactic invariant. In Retoré, C., editor, *Logical Aspects of Computational Linguistics*, pages 329–345, Berlin, Heidelberg. Springer Berlin Heidelberg.

Michel, P., Levy, O., and Neubig, G. (2019). Are sixteen heads really better than one? In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32, pages 14014–14024. Curran Associates, Inc.

Mohri, M. and Sproat, R. (2006). On a common fallacy in computational linguistics. In *A Man of Measure: Festschrift in Honour of Fred Karlsson on this 60th Birthday*, pages 432–439.

Papadimitriou, I. and Jurafsky, D. (2020). Learning Music Helps You Read: Using transfer to study linguistic structure in language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6829–6839, Online. Association for Computational Linguistics.

Partee, B. H., Meulen, A. T., and Wall, R. E. (1993). *Mathematical Methods in Linguistics*. Springer Netherlands.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison,

M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pérez, J., Marinkovic, J., and Barceló, P. (2019). On the Turing completeness of modern neural network architectures. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Pullum, G. K. (2007). Creation myths of generative grammar and the mathematics of syntactic structures. In *Proceedings of the 10th and 11th Biennial Conference on The Mathematics of Language*, MOL'07/09, page 238–254, Berlin, Heidelberg. Springer-Verlag.

Pullum, G. K. and Gazdar, G. (1982). Natural languages and context-free languages. *Linguistics and Philosophy*, 4(4):471–504.

Radzinski, D. (1991). Chinese number-names, tree adjoining languages, and mild context-sensitivity. *Computational Linguistics*, 17(3):277–300.

Salvati, S. (2015). MIX is a 2-MCFL and the word problem in $\mathbb{Z}^2$ is captured by the IO and the OI hierarchies. *Journal of Computer and System Sciences*, 81(7):1252–1277.

Seki, H., Matsumura, T., Fujii, M., and Kasami, T. (1991). On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

Shaw, P., Uszkoreit, J., and Vaswani, A. (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.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language. In *The Formal Complexity of Natural Language*, pages 320–334. Springer Netherlands.

Stabler, E. (1997). Derivational minimalism. In Retoré, C., editor, *Logical Aspects of Computational Linguistics*, pages 68–95, Berlin, Heidelberg. Springer Berlin Heidelberg.

Stabler, E. P. (2004). Varieties of crossing dependencies: structure dependence and mild context sensitivity. *Cognitive Science*, 28(5):699–720.

Stabler, E. P. (2011). Computational perspectives on Minimalism. In Boeckx, C., editor, *The Oxford Handbook of Linguistic Minimalism*, pages 617–643. Oxford University Press, Oxford.

Steedman, M. (1987). Combinatory grammars and parasitic gaps. *Natural Language & Linguistic Theory*, 5(3):403–439.

Suzgun, M., Belinkov, Y., and Shieber, S. M. (2019). On evaluating the generalization of LSTM models in formal languages. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pages 277–286.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Vijay-Shanker, K. and Weir, D. J. (1994). The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27(6):511–546.

Vijay-Shanker, K., Weir, D. J., and Joshi, A. K. (1987). Characterizing structural descriptions produced by various grammatical formalisms. In *25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Stanford, California, USA. Association for Computational Linguistics.

Weir, D. J. (1988). *Characterizing mildly context-sensitive grammar formalisms*. PhD thesis, University of Pennsylvania.

Weiss, G., Goldberg, Y., and Yahav, E. (2018). On the practical computational power of finite precision RNNs for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745, Melbourne, Australia. Association for Computational Linguistics.

Yao, S., Peng, B., Papadimitriou, C., and Narasimhan, K. (2021). Self-attention networks can process bounded hierarchical languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3770–3785, Online. Association for Computational Linguistics.