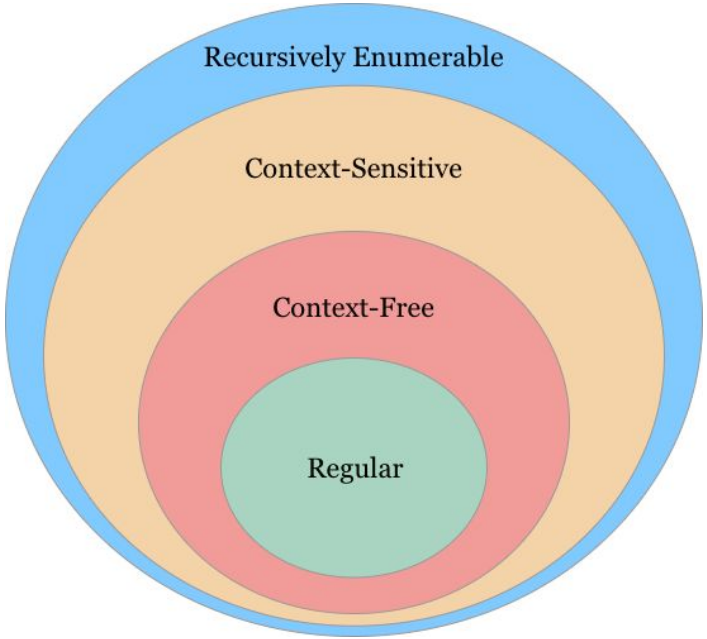
On the Ability of Neural Sequence Models to Recognize Formal Languages

Satwik Bhattamishra

Microsoft Research India

The Chomsky Hierarchy



Analysis on Formal languages

- Ability of networks such as RNNs and Transformers to recognize regular languages, context-free languages, etc.
- Provide us a controlled setting to understand a network's ability to model syntactic properties in isolation.
- Given a sequence ***s***, classify whether ***s*** belongs to a language ***L*** or not.
- Inferences can be drawn *only about syntax*.

Examples of Formal Languages

- Regular
  - Parity : 110010101 (number of ones are always even)
  - Recognizable by finite automata
  - Representable by regular expressions
- Context-free
  - $a^n b^n$
  - Dyck-1 : (( ) ( )) ( )
    - Compare the number of open open and closed parenthesis
  - Dyck-2 : [( ) [ ]] ( )
    - Also track the type of brackets

Natural and Formal Languages

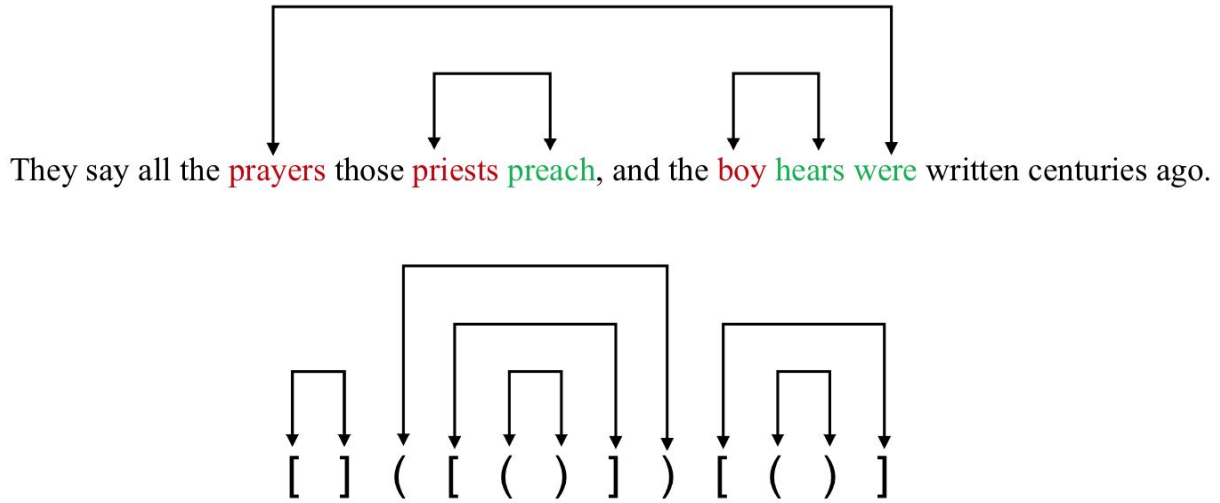


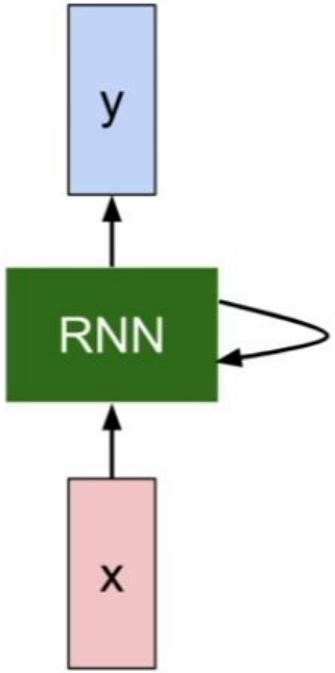
Figure 1: Nested dependencies in English sentences and in Dyck-2.

Background

Computational Power

- **Expressiveness**
- Learnability
- **Learning and Generalization in practice**

RNNs



$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

Transformer

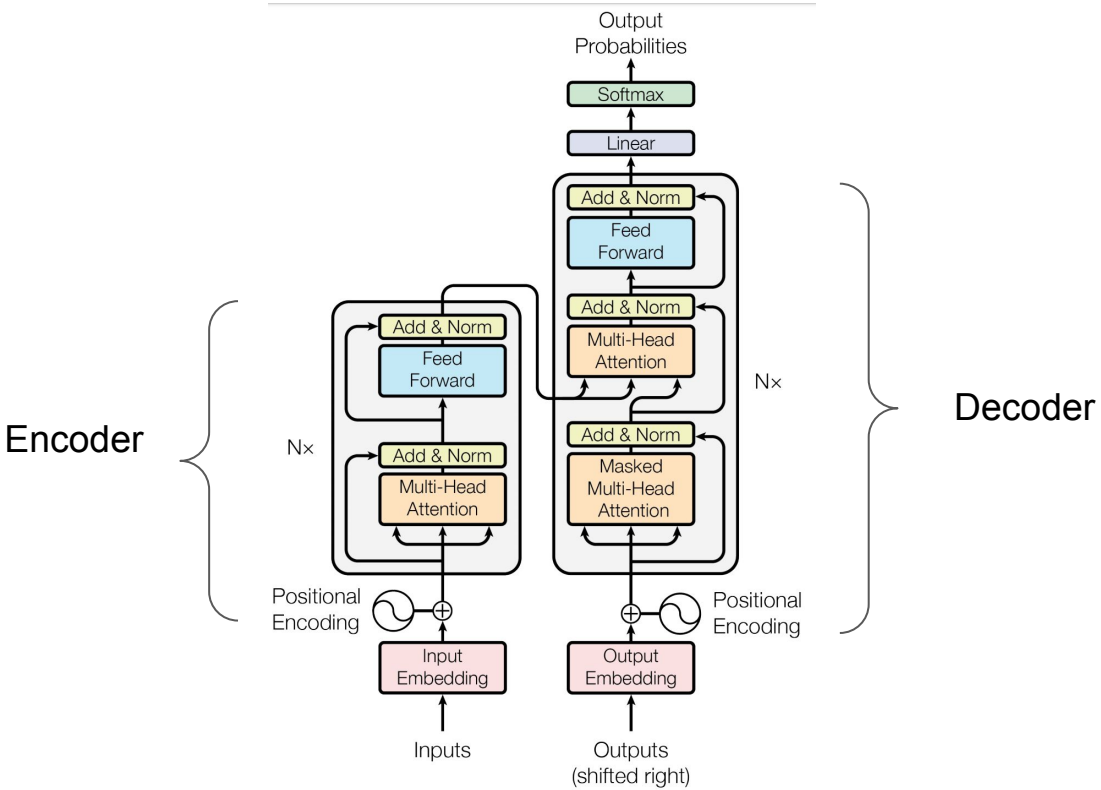


Figure 1: The Transformer - model architecture.

Results for RNNs

- On the Practical Ability of Recurrent Neural Networks to Recognize Hierarchical Languages. COLING 2020

Background: RNNs

- Work in late 20th Century
  - Different Variants - NARX, Cascade Correlation, Elman RNNs
  - TC results (Siegelmann and Sontag, 1992)
  - Regular language in finite precision (Giles, 1995)
  - Several Other works (Kolen and Kremer, 2001)

1. John F Kolen and Stefan C Kremer. 2001. A field guide to dynamical recurrent networks. John Wiley & Sons.
2. Siegelmann, Hava T., and Eduardo D. Sontag. "On the computational power of neural nets." Proceedings of the fifth annual workshop on Computational learning theory (COLT). 1992.

Recent Results for RNNs/LSTMs

- Recent Works have drawn close connections with counter automata (Merrill et al. 2020)
- Limited Performance on context-free languages (Suzgun et. al.,2019)
- Effective in modeling hierarchical dependencies in natural language data (Gulordova et. al. 2018)

1. Merrill, W., Weiss, G., Goldberg, Y., Schwartz, R., Smith, N. A., & Yahav, E. (2020). A Formal Hierarchy of RNN Architectures.
2. Suzgun, M., Gehrmann, S., Belinkov, Y. and Shieber, S.M., 2019. LSTM networks can perform dynamic counting.
3. Gulordava, K., Bojanowski, P., Grave, E., Linzen, T., & Baroni, M. (2018). Colorless green recurrent networks dream hierarchically.

Natural and Formal Languages

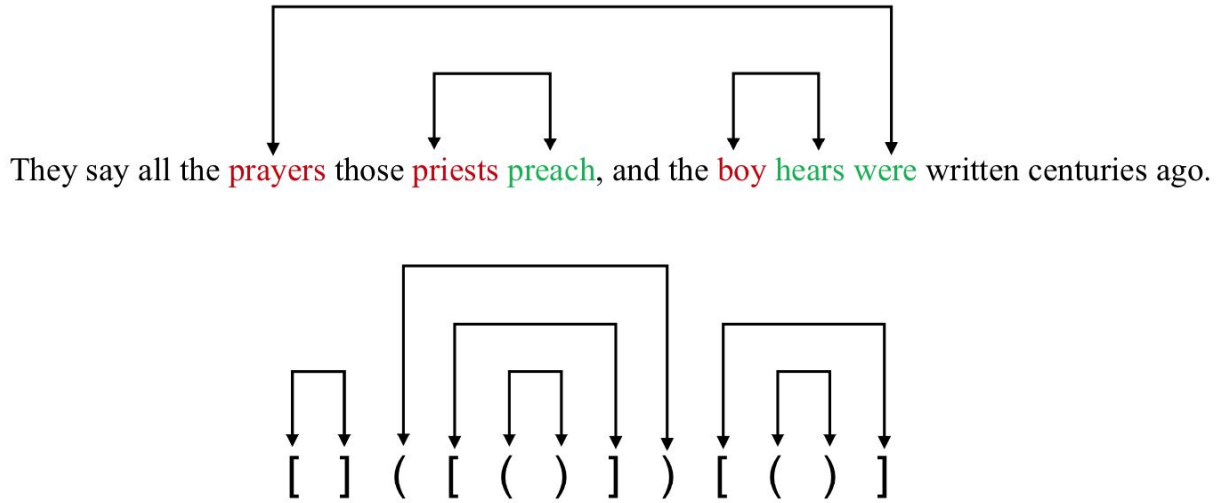


Figure 1: Nested dependencies in English sentences and in Dyck-2.

Results on Context-Free Languages

- Prior Results
  - LSTMs fail to generalize to higher lengths when trained on Dyck-2
  - On natural language data, they are able to model nested dependencies well
- Our Results
  - We consider languages with bounded-depth given humans' cognitive limitations
  - We show that RNNs are capable of recognizing Dyck-2 with bounded depth for arbitrary lengths
  - Empirical evaluation shows that they are able to learn and generalize when trained on data generated from such languages

Results: Expressiveness

Proposition 1. *Any Deterministic Pushdown Automaton can be simulated by an RNN with ReLU activation.**

- Provide a construction to show that RNNs can simulate any PDA
  - Implies that they can recognize any deterministic context-free language
- The construction implies that fixed precision RNNs are expressive enough to recognize strings of arbitrary lengths if depth of the stack is bounded

** The result holds for unbounded precision setting*

Setup

- Task: Next Character Prediction
  - For a given sequence ***s***, a model sequentially receives the input
  - At each step, the model has to predict the next set of legal/valid characters
  - During inference, a prediction is considered to be correct if and only if a model's prediction is correctly at every step
- Some previous works have also considered classification and language modeling
- Next Character Prediction task subsumes classification (recognition)
- Data
  - Train set: 10k samples
  - Validation set: 2k samples per bin

Results on bounded-depth CFLs

Language	Model	Vanilla (Randomly Sampled)		Bounded Depth		
		Bin-1A Accuracy [2, 50]↑	Bin-2A Accuracy [52, 100]↑	Bin-1B Accuracy [2, 50]↑	Bin-2B Accuracy [52, 100]↑	Bin-3B Accuracy [102, 150]↑
Dyck-2	**LSTM**	**99.5**	**75.1**	**99.9**	**99.6**	**96.0**
	Transformer	95.1	21.8	**99.9**	92.1	36.3
Dyck-3	**LSTM**	**97.3**	**54.0**	**99.7**	**96.3**	**89.5**
	Transformer	87.7	26.2	90.1	48.9	6.4
Dyck-4	**LSTM**	**97.8**	**50.7**	**99.9**	**95.1**	**87.0**
	Transformer	92.7	36.6	94.4	49.3	5.6

Results: Generalization across Depth

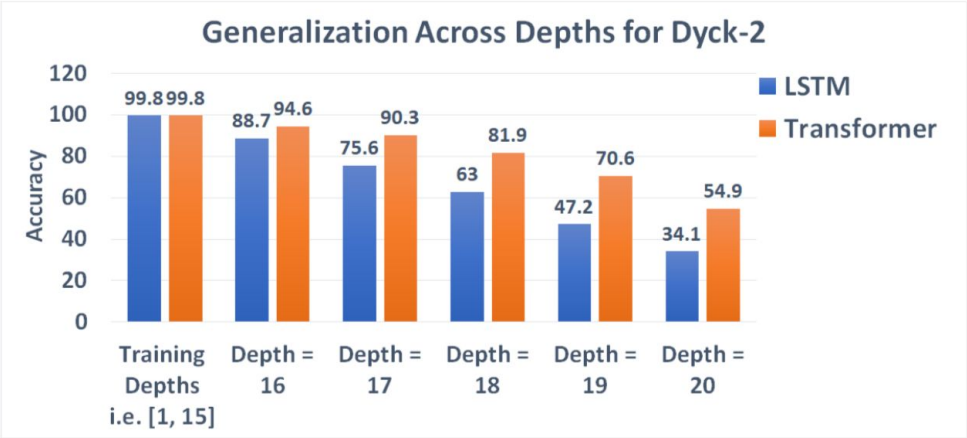


Figure 2: Generalization of LSTMs and Transformers on higher depths. The lengths of strings in the training set and all validation sets were fixed to lie between 2 to 100.

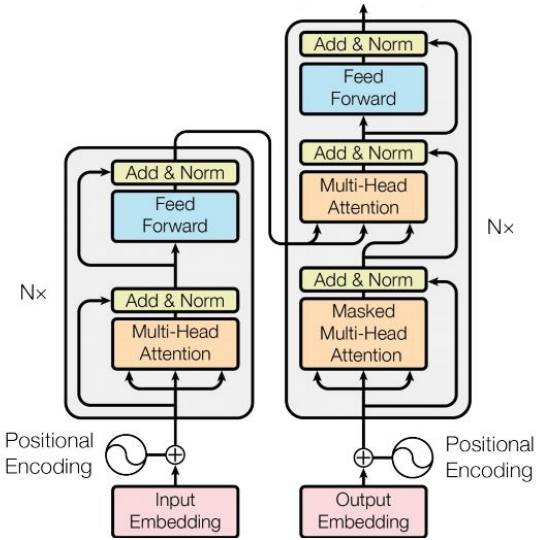
- Hewitt, J., Hahn, M., Ganguli, S., Liang, P., & Manning, C. D. RNNs can generate bounded hierarchical languages with optimal memory. EMNLP 2020
- Yao, S., Peng, B., Papadimitriou, C. and Narasimhan, K., 2021. Self-Attention Networks Can Process Bounded Hierarchical Languages. ACL 2021

Results for Transformers

- On the Ability and Limitations of Transformers to Recognize Formal Languages. EMNLP 2020
- On the Computational Power of Transformers and its Implications in Sequence Modeling. CoNLL 2020

Turing-Completeness of RNNs and Transformers

- RNNs have been shown to be Turing-complete [1]
- Recently, Transformers have also been proved to be Turing-complete [2][3]
- Compared to RNNs, Transformers consist of several components

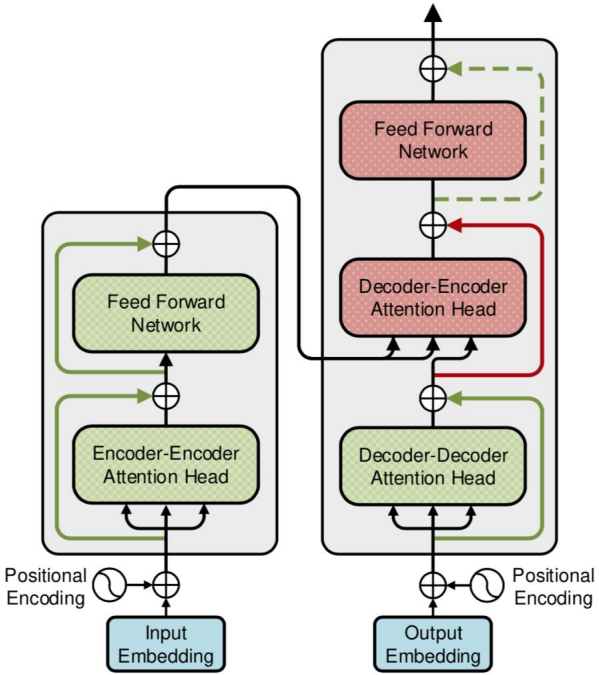


[1] Hava T Siegelmann and Eduardo D Sontag. 1992. On the computational power of neural nets. Computational learning theory, pages 440–449. ACM.

[2] Jorge Perez, Javier Marinkovic, and Pablo Barcelo. 2019. On the turing completeness of modern neural network architectures. In ICLR

[3] Satwik Bhattamishra, Arkil Patel, Navin Goyal. 2020. On the Computational Power of Transformers and its Implications in Sequence Modeling. In CoNLL

Results: Components of Transformer



The Chomsky Hierarchy

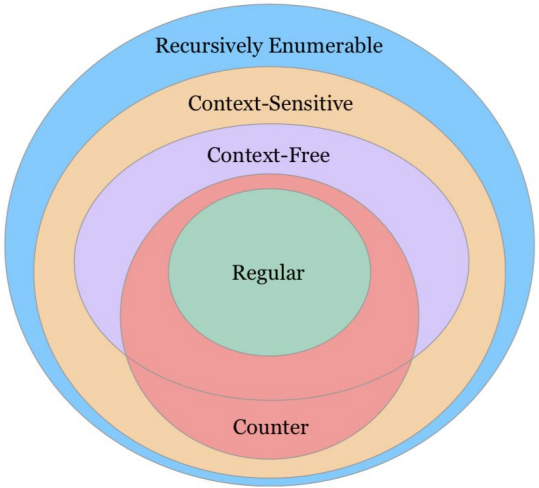


Figure 1: Counter languages form a strict superset of regular languages, and are a strict subset of context-sensitive languages. Counter and context-free languages have a nonempty intersection and neither set is contained in the other.

Counter Languages

- Dyck-1
  - Well-balanced parenthesis with a single type of bracket
  - Example: (( ) ( )) ( )
  - Although context-free, it can be solved by counting and comparing the number of open and closing brackets at each step
  - Do not necessarily need to emulate stack-like mechanism unlike Dyck-2
- Other Examples
  - $a^n b^n$
  - $a^n b^{2n}$
  - $a^n b^n c^n$

Our Results on Counter Languages

- We show that Transformers can recognize a subclass of Counter Languages
- Empirical evaluation shows that they are able to learn and generalize well on such languages
- Mechanism learned by the model is similar to our construction and hence correct

Results on Counter Languages

Language	Model	Bin-1 Accuracy [1, 50]↑	Bin-2 Accuracy [51, 100]↑	Bin-3 Accuracy [101, 150]↑
Shuffle-2	**LSTM (Baseline)**	**100.0**	**100.0**	**100.0**
	Transformer (Absolute Positional Encodings)	**100.0**	85.2	63.3
	Transformer (Relative Positional Encodings)	**100.0**	51.6	3.8
	Transformer (Only Positional Masking)	**100.0**	**100.0**	93.0
BoolExp-3	**LSTM (Baseline)**	**100.0**	**100.0**	99.7
	Transformer (Absolute Positional Encodings)	**100.0**	90.6	51.3
	Transformer (Relative Positional Encodings)	**100.0**	96.0	68.4
	Transformer (Only Positional Masking)	**100.0**	**100.0**	**99.8**
$a^n b^n c^n$	**LSTM (Baseline)**	**100.0**	**100.0**	97.8
	Transformer (Absolute Positional Encodings)	**100.0**	62.1	5.3
	Transformer (Relative Positional Encodings)	**100.0**	31.3	22.0
	Transformer (Only Positional Masking)	**100.0**	**100.0**	**100.0**

Results on Regular languages

Language	Star-Free	Transformer		LSTM	
		Bin 0	Bin 1	Bin 0	Bin 1
Tomita 1	✓	100.0	100.0	100.0	100.0
Tomita 2	✓	100.0	100.0	100.0	100.0
Tomita 3	✗	75.4	10.8	100.0	100.0
Tomita 4	✓	100.0	92.4	100.0	100.0
Tomita 5	✗	29.3	0.0	100.0	100.0
Tomita 6	✗	88.8	0.0	100.0	100.0
Tomita 7	✓	100.0	100.0	100.0	100.0

Hierarchies within regular languages

Star-free Languages

- Can be defined using regular expressions without Kleene star operator
- Cannot represent language that involve modular counting or periodicity
- Examples
  - $(ab)^* = (b\varnothing^c + \varnothing^c a + \varnothing^c aa\varnothing^c + \varnothing^c bb\varnothing^c)^c$
  - $(a(ab)^*b)^*$

Non-star-free Languages

- Examples
  - $(aa)^*$
  - $(abab)^*$
  - Parity

Results on Regular Languages

Non Star-free Languages

Star-free Languages

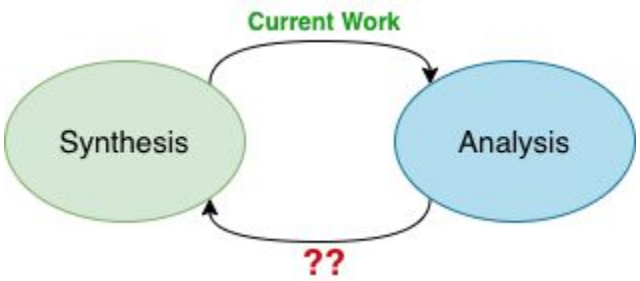
Language	Property	Transformer		LSTM	
		Bin 0	Bin 1	Bin 0	Bin 1
Parity	non-SF	68.7 (23.0)	0.0 (0.0)	100.0	100.0
$(aa)^*$	non-SF	100 (1.3)	0.0 (0.0)	100.0	100.0
$(abab)^*$	non-SF	100.0 (9.9)	5.4 (0.0)	100.0	100.0
$\mathcal{D}_1$	depth-1	100.0	100.0	100.0	100.0
$\mathcal{D}_2$	depth-2	74.6	3.1	100.0	100.0
$\mathcal{D}_4$	depth -4	90.2	3.3	100.0	100.0

Takeaways

- RNNs
  - Very powerful in both finite and infinite precision settings
  - In Finite Precision: Regular language
  - Context-free and Counter Languages to a certain extent
- Transformers (Encoder only model)
  - Can recognize a subclass of counter languages such as Dyck-1, n-ary expression
  - But are limited in recognizing certain regular languages such as non-star-free and star-free with high dot-depth
  - Overall - Clearly limited compared to RNNs/LSTMs

Questions

- Why do Transformers perform so well in practice despite limitations?
  - Efficiency in Training?
  - Less sensitive towards single input change
  - Less prone to overfitting at scale
  - Capturing long range dependencies better
- Implications on programming languages



Thank You