Natural Language Processing Final Project
Project 4: Modern Neural Networks Meet Linguistic Theory

First Author **Second Author** **Third Author**

{Ayça Avcı, Behrooz Nikandish, Pooja Gowda}@student.rug.nl

University of Groningen
Groningen, The Netherlands

Abstract

This project provides a scope to understand how Natural Language Processing helps advance cognitive modeling? To do so, we replicated the results from a study by Kirov and Cotterell (2018) and concentrated on the learn-ability of linguistic rules in the context of past tense verbal inflection with research directions. The first research question is to explore *if the insights by Kirov and Cotterell stand when using a transformer model*. Another research question is *to find how the amount of available data affects the learning of irregular past tense forms?*. The modern Neural Networks (NNs) for cognitive modeling are used to train the neural network to transduce the English verb stems to their past tense forms. A Long Short-Term Memory Networks (LSTM) encoder-decoder architecture learns both regular and irregular past tense formation in the observed data and generalizes reasonably to held-out verbs. For the research directions, we experimented with the available data and monitored how it affects the learning of irregular past tense forms. Also, we employed the transformer model to replicate the experiment by Kirov and Cotterell to discuss the insights behavior. As a result of replication of experiment 1 and experiment 2, the LSTM encoder-decoder architecture was trained with an accuracy of 99%, validation accuracy of 99.18%, and test 99.26 for single-task all verbs and with similar results for multi-task all verbs. On another hand, by changing the data size we observed effects on the learning ability of irregular past tense forms due to a decrease in validation accuracy and test accuracy. We conclude the project by discussing the resultant changes in the accuracy, also indicating the possible reason for this. You can find the github repository here: (`https://github.com/aycavci/Natural-Language-Processing`)

1 Introduction

In 1986, Rumelhart and McClelland (R&M) illustrated how a neural network competent in transducing English verbs stems from their past tense in the opus. This illustration was also been exposed to strong criticism raised by Pinker and Prince in 1988 (P&P) (Pinker and Prince, 1988). As per their criticism, they stressed the poor empirical performance of this model and indicated a number of issues with respect to theoretical aspects of the model.

NLP is a domain with many unknown architectures, a large size of data sets, and the availability of abundant computational resources provides scope for the state of art applications. On another hand, this field exhibits to have a practical disposition. Hence, using engineering applications of language technologies makes it very difficult to be considered. There are many recent works, indicating the perception change, for instance, studying the capability of neural networks to learn syntactic dependencies such as subject-verb agreement (Tal Linzen, 2016).

The R&M model aims to generate inflections from data, for example., generating English past tense forms from their verb stems. But, this model failed to generalize accurately during the testing phase. On another hand, using morphological generation networks built from modern Recurrent Neural Networks (RNNs) provides greater results in generalization (Ryan Cotterell and Hulden, 2016). The results suggested and bring light to considering neural networks with Natural language and linguistic science.

(Kirov and Cotterell, 2018) generalized the RM setting by requiring the study of more mappings than just a lemma to past tense and they worked specially on a issue that was part debate whether or not neural models learn and use "rules". The goal would be to map a lemma, such as walk, to its past-tense word walked, as well as its gerund and third person present singular forms, walking and walks.
The researcher here focused on two research questions:

1. *Does the learner induce the full set of correct generalizations about the data? Given a range of novel inputs, to what extent does it apply the correct transformations to them?*
2. *Does the behavior of the learner mimic humans? Are the errors human-like?*

The work is carried out by performing experiments that examine the encoder-decoder architecture's ability in machine translation. On the replication of these experiments, we did get results of good training accuracy of 99.46% and with testing results with prediction average score of -0.0032 for experiment 1, 99.62% and prediction average score of -0.0032 for experiment 2. And the model learning did mimic the behavior of the learner where they try to form past tense from regular verbs and sometimes they over regularise the verbs. This project's research direction that we worked on are follows:

1. *How does the amount of available data affect the learning of irregular past tense forms? Is there a threshold in the amount of data, or in the ratio between regular and irregular forms, that hinders the learning process?*

2. *LSTMs process sequences in a left-to-right fashion, and because of this have been considered more "principled" than Transformer models for the processing of natural language. This did not prevent Transformer to replace recurrent architectures and become ubiquitous in NLP. Do the insights by Kirov and Cotterell (2018) stand when using transformer-based models?*

In the research direction of data experimentation, We experimented with different data size and documented the results. The results suggest that the learning task of irregular past tense form is affected when we test with *tgt_test.txt*. Train accuracy remains to be around 98-99%, while the test accuracy and validation accuracy reduces drastically as we reduce the regular verbs data size. In the other research direction, we replicated the model using Transformer architecture (Vaswani et al., 2017). The model performed admirably during the training and validation steps with an accuracy of around 97-98%. However, even after optimizing the hyper-parameters for low-resource conditions (Araabi and Monz, 2020), the performance of the model was unsatisfactory when testing with new data.

1.1 The Team

Pooja Gowda is a master's student pursuing a Computing Science program (Data science and System Complexity), she has a keen interest in working with machine learning and data science methods, she worked on replicating the model worked by Kirov and Cotterell and data experimentation research direction and writing the introduction, discussion of results and conclusion.

Behrooz Nikandish is a master's student in Information Science, interested in data science and data analytics with Python. He contributed to replicating the Kirov and Cotterell code in Python3. He mainly worked on the first direction to replicate the project using transformers and writing the methods, and part of the results and discussion.

Ayça Avcı is a master's student in Computing Science: Data Science & Systems Complexity. She is interested in machine learning and data science. She worked on a second research direction and replicated the results from Kirov and Cotterell paper. She wrote experimental setup, and, replication of experiment in paper of results and discussion.

2 Method

2.1 Encoder-Decoder Architecture

The Encoder-Decoder network architecture (ED), or sequence to sequence network is a common Machine Translation (MT) technique that can be implemented with RNNs or Transformers. The utilization of an encoder network to take an input sequence and construct a contextualized representation of it, referred to as the context, is the core notion behind these networks. After that, the representation is sent to a decoder, which produces a task-specific output sequence. ED architecture can be implemented using LSTMs, Convolutional Neural Networks (CNNs), and Transformers (Jurafsky and Martin, 2021). Figure 1 shows the basic architecture of encoder-decoder networks.

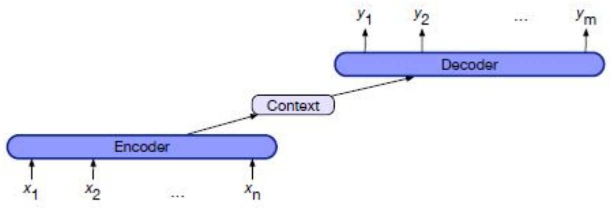


Figure 1: The architecture of the encoder-decoder. The context, which is a function of the input's hidden representations can be used by decoder (Jurafsky and Martin, 2021).

2.2 ED Architecture using RNNs

In the vanilla recurrent neural network (RNN) of ED architecture, the decoder employs context information from the encoder's last hidden state.

The source and target sentences are concatenated with a separator token in between. An attention mechanism can play the role of a bottleneck solution here that allows the decoder to acquire information from all of the encoder's hidden states, not just the last one (Jurafsky and Martin, 2021).

The ED architecture that (Kirov and Cotterell, 2018) employed in their project comprises two RNNs linked by an attention mechanism. Each symbol in the input string is read one at a time by the encoder RNN, which assigns it a unique embedding before processing it to construct a representation of the phoneme given the remainder of the phonemes in the text. The decoder RNN generates a series of output phonemes, employing the attention mechanism to return to the encoder states as needed.

2.3 ED Architecture using Transformers

The transformer is the first transduction model that computes its input and output representations using only self-attention rather than RNNs. The encoder converts a sequence of symbol representations into a continuous representation sequence. After that, the decoder creates a symbol output sequence, one element at a time. At each step, the model utilizes the previously created symbols as extra input to create the next (Vaswani et al., 2017).

Figure 2 shows the architecture of transformer model. For the encoder-decoder architec-

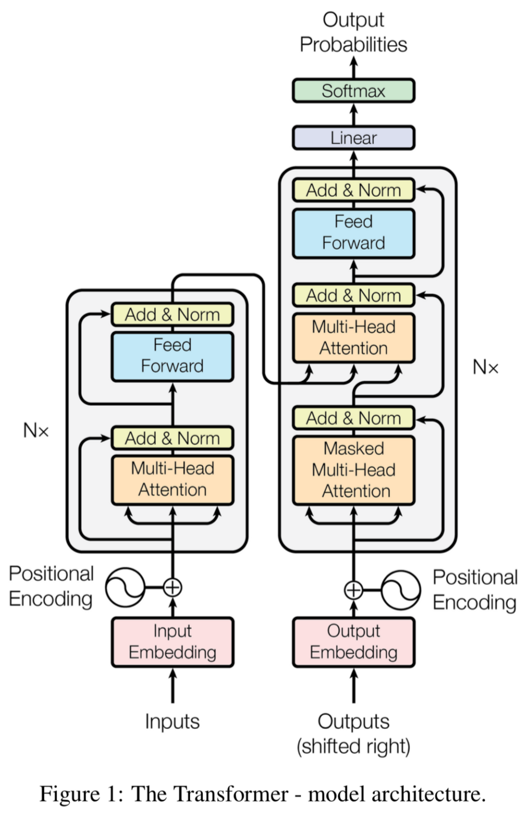


Figure 2: The architecture of the Transformer model (Vaswani et al., 2017).

ture, a transformer could be employed. It comprises an encoder that maps the input words from the source language to an output representation. The decoder is a conditional language model that attends to the encoder representation and generates target words one by one, based on the source word and previously generated target language words, at each time-step.

Kirov and Cotterell developed an LSTM-based model in 2018. This project employed LSTMs and transformers separately to reproduce their model for learning the English past tense.

3 Experimental Setup

3.1 Replicating results from Kirov and Cotterell, 2018

3.1.1 Experiment 1

In this experiment, we tried to replicate results from (Kirov and Cotterell, 2018). The aim of this experiment is to train an LSTM model that learns conjugate of both regular and irregular verbs in English language. To achieve this, we used CELEX (Baayen et al., 1993) data set which contains 4039 verbs. Each verb appears only once in the data set, meaning that data is uniformly distributed. 168 of them marked as irregular and the rest marked as regular. Data set looks like Table 1.

	present tense form	past tense form	present tense IPA	past tense IPA	label
0	ventilate	ventilated	vEnt@leIt	vEnt@leItId	reg
1	forget	forgot	f@gEt	f@gOt	irreg

Table 1: Representation of CELEX data set. IPA refers to the International Phonetic Alphabet.

Data set divided into train (80%), validation (10%), and test (10%) sets after shuffling. OpenNMT (Klein et al., 2017) package is used for the experimentation. to Each word embedding has a size of 300 units. Encoder is set to bidirectional LSTM (Hochreiter and Schmidhuber, 1997), and decoder is set to unidirectional LSTM, both having two layers. 0.3 dropout applied between the layers. Both the encoder and decoder contains 100 hidden units. Adadelta (Zeiler, 2012) used as an optimizer, and training is done with learning rate 1.0 and batch size 20 for 100 epochs (1 epoch correspond to 50 train_step since the vocabulary generated by 1000 examples/corpus). Model is decoded with beam search with a default value of k=12. Accuracy as used as an performance evaluation metric of the model. All the parameters used can be found below:

- *rnn_type:* LSTM (by default)
- *rnn_size:* 100
- *train_steps:* 5000
- *learning_rate_decay:* 1.0
- *learning_rate:* 1.0 (by default)
- *word_vec_size:* 300
- *encoder_type:* brnn
- *decoder_type:* rnn (by default)
- *enc_layers:* 2 (by default)
- *dec_layers:* 2 (by default)
- *dropout:* 0.3 (by default)
- *optim:* adadelta
- *batch_size:* 20

We calculated the model accuracy on test set using prediction score, which is the log likelihood of the generated sequence. Equation 1 represents the calculation of the likelihood. Likelihood is the probability of the generated sequence, multiplying it with 100 gives the accuracy percentage.

$$Pred\ avg\ score = \log_{10}(likelihood) \quad (1)$$

We will treat model generated in (Kirov and Cotterell, 2018) as baseline. We will compare our findings with regards to baseline model.

3.1.2 Experiment 2

In this experiment, gerund, past participle and third-person singular forms are included to the CELEX data set that is used in Experiment 1 according to corresponding labels in Wiktionary (Sylak-Glassman et al., 2015). Model is trained on each stem-inflection pairs. Its input is modified with characters which represents transduction such as *get <PST> -> got* and *get <PTCP> -> gotten*. Model is trained with the same architecture and parameters used in Experiment 1 to jointly predict the mappings of English verbs (past, gerund, past participle, third-person singular).

3.2 Experimenting with Transformers

Experiment 1 is repeated using the Transformers model instead of LSTM to compare both model performances on CELEX data set. For this experiment, we first modified default parameters to the optimal settings for low-resource data conditions (Araabi and Monz, 2020) due to the size of the data set. Parameters used for Transformer model setting can be found below:

- *decoder_type:* transformer
- *encoder_type:* transformer
- *word_vec_size:* 512
- *feature_vec_size:* 512
- *rnn_size:* 512
- *layers:* 5
- *transformer_ff:* 512
- *heads:* 2
- *accum_count:* 4
- *optim:* adam
- *adam_beta1:* 0.9
- *adam_beta2:* 0.998
- *decay_method:* noam
- *learning_rate:* 0.1
- *batch_size:* 32
- *dropout:* 0.3
- *label_smoothing:* 0.6

3.3 Experimenting with different data set size

Experiment 1 is repeated with different sizes of regular and irregular verbs within CELEX data set. Same settings are used as described in Experiment 1. Distribution of the regular and irregular verbs in the data set can be seen in Table 3.

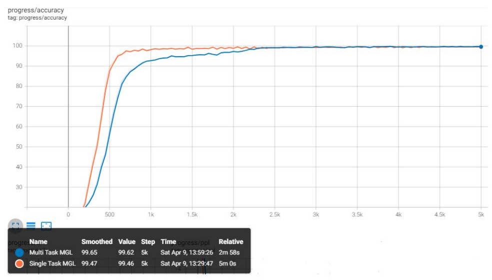
4 Results & Discussion

4.1 Replicating results from Kirov and Cotterell, 2018

Figure 3 and 4 represents the model accuracy change for both training and validation set respectively during each training steps. Training accuracy is converged for both single and multi-task to 99.46% and 99.62% respectively when the training step is 2.25K. Validation accuracy is converged for both single and multi-task to 99.18% and 99.96% respectively when the training step is 1K.

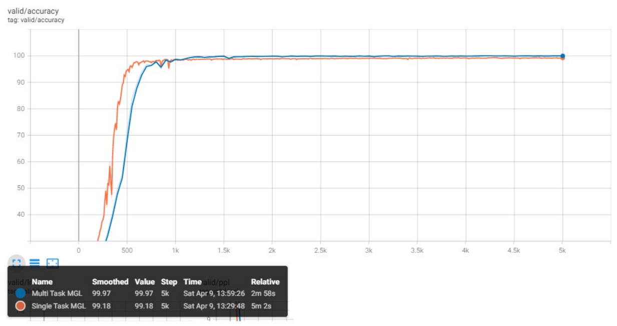
	train	val	test	pred avg score	pred ppl
Single-Task All Verbs	99.46	99.18	99.26	-0.0032	1.0032
Multi-Task All Verbs	99.62	99.96	98.87	-0.0049	1.0049

Table 2: Train accuracy (in %), val accuracy (in %), test accuracy (in %), prediction average score (pred avg score) and prediction perplexity (pred ppl = exp(-pred avg score)) for both single and multi-task models for all verbs (both regular and irregular ones).



	Name	Smoothed	Value	Step	Time	Relative
●	Multi Task MGL	99.65	99.62	5k	Sat Apr 9, 13:59:26	2m 58s
●	Single Task MGL	99.47	99.46	5k	Sat Apr 9, 13:29:47	5m 0s

Figure 3: Training accuracy (in %) for both models from experiment 1 (single-task) and 2 (multi-task). x-axis corresponds to the training steps and y-axis corresponds to the training accuracy.



	Name	Smoothed	Value	Step	Time	Relative
●	Multi Task MGL	99.97	99.97	5k	Sat Apr 9, 13:59:26	2m 58s
●	Single Task MGL	99.18	99.18	5k	Sat Apr 9, 13:29:48	5m 2s

Figure 4: Validation (val) accuracy (in %) for both models from experiment 1 (single-task) and 2 (multi-task). x-axis corresponds to the training steps and y-axis corresponds to the val accuracy.

Table 2 shows that ED model on both single-task and multi-task is able to learn past tense conjugates and multiple types of mappings of English verbs (both regular and irregular ones) successfully with 99.26% and 98.87% test set accuracy respectively. When compared Table 3 in (Kirov and Cotterell, 2018), our model has slightly less training accuracy (less than 0.5% difference), meanwhile having slightly higher accuracy on val and test set (between 2%-5% difference) on both task.

In Figure 3 and 4, we can see that single-task has a steeper learning curve than the multi-task compared to Figure 1 in (Kirov and Cotterell, 2018). this means that same level of performance achieved more quickly on single-task compared to multi-task. However, final accuracy on both training and val set for multi-task model is slightly higher than the single-task one. It is evident that LSTM model learns multiple mappings slower since it generates the mappings jointly, but it results higher accuracy at the end. This might facilitate single-task learning due to shared pattern in phonology.

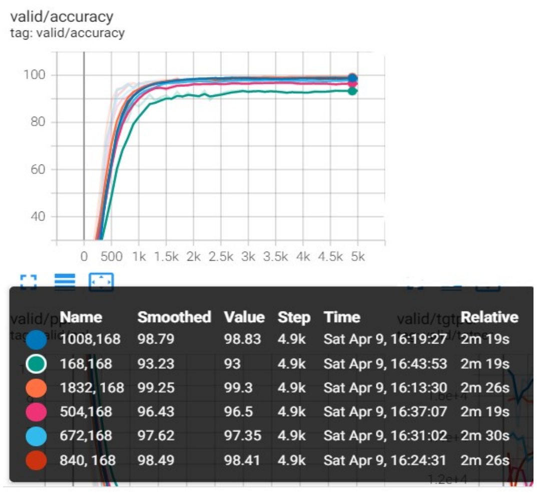
After error analysis on both tasks, we saw that the model only makes the error on irregular verbs by overregularizing them (e.g., throw -> throwed). It can be explain that humans also have a tendency to treat some irregular verbs that are unknown to them as regular ones. Since it is due to the lack knowledge on irregularity, we can conclude that our model demonstrates human-like error behaviour, which is acceptable.

4.2 Direction 1 - Data Experimentation

In this research direction, we experimented using the replicated LSTM model of (Kirov and Cotterell, 2018). Here, we tested the learning ability of irregular past tense forms with different data sizes.

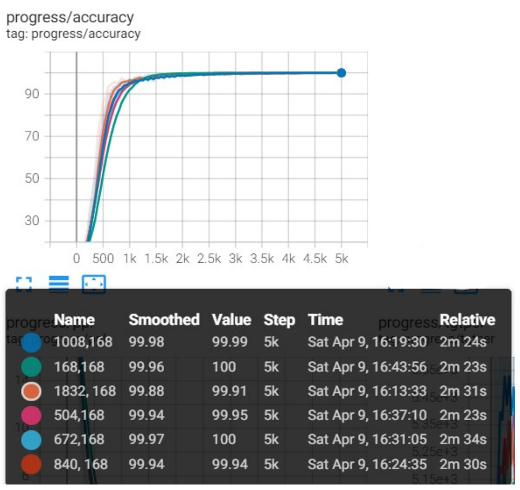
Reg	Irreg	Train(%)	Dev(%)	Test(%)
1832	168	99.79	99.241	98.855
1008	168	99.99	98.831	97.701
840	168	99.94	97.558	97.656
672	168	100.00	97.9381	97.656
504	168	99.95	96.3107	98.764
168	168	100.00	93.7743	92.853

Table 3: Accuracy with respect to various data size (Single-task). For each data size distribution the resultant train, validation and test accuracy percentages for different size of regular and irregular verbs in the data set.



	Name	Smoothed	Value	Step	Time	Relative
●	1008,168	98.79	98.83	4.9k	Sat Apr 9, 16:19:27	2m 19s
○	168,168	93.23	93	4.9k	Sat Apr 9, 16:43:53	2m 19s
●	1832, 168	99.25	99.3	4.9k	Sat Apr 9, 16:13:30	2m 26s
●	504,168	96.43	96.5	4.9k	Sat Apr 9, 16:37:07	2m 19s
●	672,168	97.62	97.35	4.9k	Sat Apr 9, 16:31:02	2m 30s
●	840, 168	98.49	98.41	4.9k	Sat Apr 9, 16:24:31	2m 26s

Figure 5: Validation accuracy affected for different data size. x-axis represents step size and y-axis represents accuracy in percentage.



	Name	Smoothed	Value	Step	Time	Relative
●	1008,168	99.98	99.99	5k	Sat Apr 9, 16:19:30	2m 24s
●	168,168	99.96	100	5k	Sat Apr 9, 16:43:56	2m 23s
○	1832, 168	99.88	99.91	5k	Sat Apr 9, 16:13:33	2m 31s
●	504,168	99.94	99.95	5k	Sat Apr 9, 16:37:10	2m 23s
●	672,168	99.97	100	5k	Sat Apr 9, 16:31:05	2m 34s
●	840, 168	99.94	99.94	5k	Sat Apr 9, 16:24:35	2m 30s

Figure 6: Training accuracy affected for different data size. x-axis represents step size and y-axis represents accuracy in percentage.

We trained the model with different data sizes and compared the train, validation, and test accuracy concerning every data size change. As per the Table 3, we can say that as we reduce the regular verb data size keeping the irregular data size constant, the test and validation accuracy rates are affected, as it gradually decreases. While the training accuracy remains unchanged.

The data size distribution of irregular verbs is kept constant (168) and we gradually reduced the regular verbs. On reduction of the data size, we obtained the results given in Table 3. As per our observation, the test accuracy is affected since the trained model is tested using *tgt_test.txt*, this file is generated randomly with a random distribution of irregular and regular verbs. Hence, it can be a possible cause to obtain a gradual reduction in test accuracy and validation accuracy. Figure 5 is the validation accuracy graph plotted and Figure 6 is the progress accuracy graph plotted.

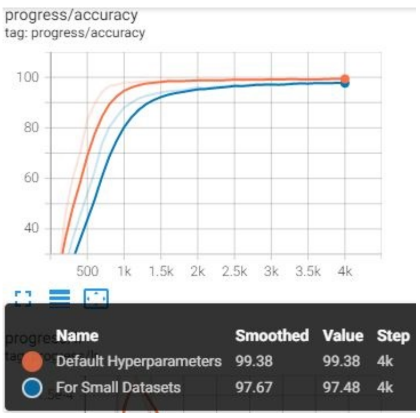
4.3 Direction 2- Transformers

According to our findings, the single-task model performs effectively throughout the training stage. Table 4 compares the accuracy of our transformer-based model to (Kirov and Cotterell, 2018) in the training and validation phase.

architecture	train	dev
Transformer	97.48	98.5395
K&C LSTM	99.46	99.18

Table 4: Comparison of training accuracy and validation accuracy of transformer-based model and Kirov and Cotterell's LSTM-based model in English past tense prediction.

Figure 7 shows the progress accuracy for experiments in default settings and optimal settings for small datasets.

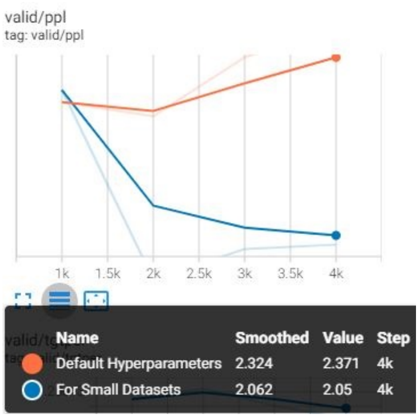


	Name	Smoothed	Value	Step
●	Default Hyperparameters	99.38	99.38	4k
●	For Small Datasets	97.67	97.48	4k

Figure 7: Training accuracy using Transformers.

Despite having no built-in knowledge of phonology, the ED model utilizing transformers effectively learns to conjugate verbs nearly perfect in the training stage, including irregulars.

Figure 8 shows the improvement of the val-



	Name	Smoothed	Value	Step
●	Default Hyperparameters	2.324	2.371	4k
●	For Small Datasets	2.062	2.05	4k

Figure 8: Validation perplexity observed using transformers with default and optimal hyper-parameters.

idation perplexity by optimizing the hyper-parameters. The importance of the proper hyper-parameters is revealed after testing the trained model. The prediction average score improved from -2.1826 using the default setting to -0.8695 in the optimal setting. Similarly, the prediction perplexity dropped from 8.8696 with default hyper-parameters to 2.3858 with optimized hyper-parameters. However, in testing the model with new unseen data, LSTM-based model outperformed Transformer. Table 5 compares the results of our experiments in testing the trained model using LSTMs and Transformers. The reason is mainly down to the fact that Transformer-based models need large data sets to learn effectively. "Optimized Transformer only outperforms LSTM with more than 20k training examples." (Araabi and Monz, 2020).

Model	PRED AVG SCORE	PRED PPL
Transformer	-0.8695	2.3858
LSTM	-0.0032	1.0032

Table 5: Comparison of prediction average score and prediction perplexity of our experiments using Transformer and LSTM architectures in single task experiment with all verbs.

5 Conclusions

This project aimed to replicate the model of (Kirov and Cotterell, 2018). We were able to replicate the results successfully. ED model performs similarly both on single and multi-task compared to (Kirov and Cotterell, 2018). Since ED model shows human-like performance, it can used as a research tool in linguistics and NLP. Using the replicated model, we explore around with research direction 1. Here, we find out how the change in data size distribution affects the learning of irregular past tense forms. To explore this direction, we ran experiment 1 from the replicated model of (Kirov and Cotterell, 2018) without making any changes to parameters, and we kept the irregular verbs data size constant and gradually decreased the data size regularly to observe how it affects the learning task. The results demonstrated a gradual decrease in test and validation accuracy, while the training accuracy remained unaffected and constant (98-99%). From these insights, we concluded our research direction1 speculating that this behavior cause is due to the randomly created tgt_test.txt file with an unknown distribution ratio of irregular and regular verbs. We also used transformers to replicate the model as part of the second research direction. We used this model to run Experiment 1 to see how transformer architecture affects the outcomes. The model worked practically smooth during the training and validation steps. However, When testing the model using previously unseen data, the results were unsatisfactory. Our findings show under extremely low-resource data RNN-based models still outperform Transformer-based ones (Araabi and Monz, 2020).

References

Ali Araabi and Christof Monz. 2020. Optimizing transformer for low-resource neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3429–3435, Barcelona, Spain (Online). International Committee on Computational Linguistics.

R Harald Baayen, Richard Piepenbrock, and H Van Rijn. 1993. The celex lexical database (cd-rom). linguistic data consortium. *Philadelphia, PA: University of Pennsylvania*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Dan Jurafsky and James H. Martin. 2021. *Speech and Language Processing (3rd ed. draft)*.

Christo Kirov and Ryan Cotterell. 2018. Recurrent neural networks in linguistic theory: Revisiting pinker and prince (1988) and the past tense debate.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation.

Steven Pinker and Alan Prince. 1988. On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition*, 28(1):73–193.

John SylakGlassman David Yarowsky Jason Eisner Ryan Cotterell, Christo Kirov and Mans Hulden. 2016. The sigmorphon 2016 shared task–morphological reinflection. In *In Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22.

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A language-independent feature schema for inflectional morphology. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 674–680, Beijing, China. Association for Computational Linguistics.

Yoav Goldberg Tal Linzen, Emmanuel Dupoux. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics (TACL)*, 4:521–535.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Computing Research Repository*, arXiv:1706.03762. Version 2.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method.