
A Reversible Transformer based Bangla conversational agent

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 Chatbots are becoming more popular on many industrial platforms as AI
2 advances and the need for an online discussion system grows. Intelligent
3 tools are taking attention of their applications since they can emulate hu-
4 man behavior in natural languages. Their use is becoming more fluid and
5 simpler over time. This is due to advancements in the disciplines of natural
6 language processing and AI. In this article, we attempt to present a model
7 based on Transformer and the framework Trax for Bangla chatbot. We
8 conclude with a discussion that contrasts all of the strategies.

9 1 Introduction

10 Artificial intelligence (AI) has paved the way for new technological achievements by flour-
11 ishing computer power. AI's most significant application is Natural Language Processing
12 (NLP). It is a method of making the machine or a computer understand the human language
13 (1). One of the prominent applications of NLP is conversational agents (2), commonly known
14 as chatbots in the media and industry. Even though FAQs are a regularly used method,
15 chatbots have gained popularity due to their high responsiveness (8). A chatbot is an AI
16 based software that simulates and analyzes human dialogue, whether written or spoken
17 (3). A chatbot offers the end-user with 24-hour availability, quick replies, infinite patience,
18 personalization, and customizable discourse (5). In future it is expected that Chabot will
19 decrease the workload up to 70% (4). In the AI field, developing intelligent conversational
20 bots is still an unsolved research subject that poses several hurdles. We are attempting to
21 identify a new algorithm for building chatbots via this effort.

22 In the English language, there are a number of natural language chat systems that may
23 be used in a variety of situations. However, there is no noteworthy study or research for a
24 man-machine communication system in Bangla (6). Bangla is the 6th most widely spoken
25 language. Therefore, we attempted to make a model that can be implemented with various
26 language based on its dataset and implemented with Bangla language.

27 Transformer model is one of the newest techniques of NLP. However, it has issues of memory
28 and inference time when it handles long sequences (10). Chatbot uses long sequence models,
29 hence, we come up with the new model that is reversible transformer to implement the
30 chatbot. Trax is the neural network platform as Keras but have the ability to work faster
31 and quite new to this field. Therefore, our model is based on new technology and mitigates
32 all the problems of transformer model.

33 The purpose of NLP is to take unstructured data and turn it into a structured text represen-
34 tation with comprehensible language for textual chatbot conversations. The description of

our framework and the chatbot technique employed in this article are then shown, together with experimental results.

2 Literature Review

Acknowledging the potential of Bangla conversational agent the authors of (6) emerged with a man-machine conversational system in Bengali language named "TUNI". They employed Bangla Natural Language Processing (BNLP) to create this chatbot, which behaves as a psychologist and asks questions. A rule-based method was used to extract the input information and map them with output. They solved Bangla Pos Tagging, Suffix Processing, Person Mapping, PosNeg Mapping, Sentence Deviation Processing as a pre-requisite of solving their model. Finally, when compared to the benchmark English conversational agent ELIZA, TUNI outperformed.

Chatbots have excelled in the realm of healthcare, although they are confined to the English language. As a solution, the authors of (7) created "DISHA," a machine-learning-based closed domain healthcare Bengali language chatbot. They employed a text-based method in which the system reads written language and analyses it using the Named Entity Recognition (NER) algorithm. The client's name, blood group, age, and other information were extracted using a modified version of this algorithm. For their independent Bangla disease classification dataset they used independent test set and K-fold cross validation (CV) and achieved a success rate of 97.26% for Decision Tree, 97.82% for random forest, 95.73% for Multinomial NB, 98.39% for SVM, 97.42% for AdaBoost and 96.13% for KNN.

A combination of formal language theory and natural language understanding for chatbot implementation has been shown in the research (9). They have used context-free grammar to optimize their algorithm. The training and testing data has been collected from Romanian language. Finally, they came up with a solution of error validation by implementing learning strategy.

3 Model Architecture

3.1 Multi-head Attention

To run fast and get a good result, this model uses multi-head attention (MA). It runs parallel attention mechanism of the matrices multiple time each called the heads in the model. The first head uses a sets of representation and the second one implements linear transforming of the original embedding by using set of matrices. Each head uses different linear transformation to represent words that is different heads learns different relationships between words. The original word embedding is been multiplied with K, Q, V matrix to get the corresponding keys, queries and values. As a result, one head can understand the relationship between words from another head. The initial embedded word has multiplied by Q to get q_1 , K to get k_1 and V to get v_1 . Then it has been fed to the linear layer and the score has been calculated to get the probability and multiplied by the value to get a new representation of the word. Each word then multiplied by the corresponding W_Q, W_K, W_V to get the embedding. Then the scores has been calculated as follows:

$$Z = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

To get the final score, all the Z has been concatenated and multiplied with W_O as follows:

$$Z = Concat(Z_1 Z_2 \dots Z_n) \times W_o \quad (2)$$

3.2 Locality Sensitive Hashing

The locality sensitive hashing (LSH) is been computed for the word vector (11). The transformed word embedding corresponds to the vector generating transformed outputs. The

MA is used for the measure of the similarity of query and the key. The LSH first compute the nearest neighbor to q among vectors k_1, \dots, k_n . Attention computes $d(q, k_i)$ for i from 1 to n . For every q is close to k_i it checks $hash(q) == hash(k_i)$. Then random cutting space is calculated with the function below:

$$hash(x) = sign(xR) \quad R : [d, n_hash_bins] \quad (3)$$

After hashing Q and K the standard attention has been computed on the bins that has created. The MA will repeat the same process several times that increased the probability to get the same key in the bin as the query. For the LSH attention used in this paper for a single query position i at particular time the equation goes as follows:

$$\alpha_i = \sum_{j \in \mathcal{P}_i} e^{(q_i \cdot k_j - \mathcal{Z}(i, \mathcal{P}_i))} v_j; \quad \mathcal{P}_i = j : i \geq j \quad (4)$$

Here, (\mathcal{P}_i) is the set of query at i position attends and (\mathcal{Z}) is the partition function. For batching, the attention over large set has performed as $\widehat{\mathcal{P}}_i = 0, 1, \dots, l \supseteq \mathcal{P}_i$ with masking the elements out not in (\mathcal{P}_i)

$$\alpha_i = \sum_{j \in \mathcal{P}_i} e^{(q_i \cdot k_j - m(j, \mathcal{P}_i) - \mathcal{Z}(i, \mathcal{P}_i))} v_j; \quad m(j, \mathcal{P}_i) = \begin{cases} \infty & \text{if } j \notin \mathcal{P}_i \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

The sequence of the query and the keys then hashed into the bucket and sorted after bucket. The bucket then split into chunks for parallel computing. Then the MA within the same bucket of the chunk is calculated looking at previous chunk.

3.3 Reversible Residual Layers

The biggest issue with the transformer mechanism is of time and memory complexity. For handling long sequences (LS) it requires a lot of memory thus time to compute. However, to compute LS it does not require to consider every L position rather it is good to consider only an area of interest. By the use of MA a single word and words immediately around it can be focused and recompute the activations that saves memory and reduce time. To overcome the situation this model uses reversible residual layers (RRL).

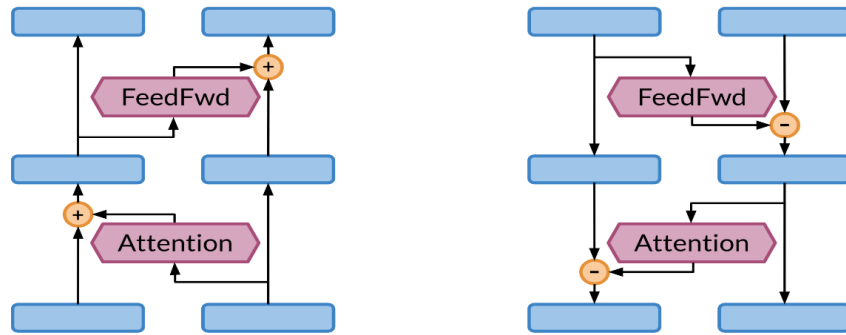


Figure 1: Reversible Residual Layers Back Propagation

RRL allows to reconstruct the forward layer from the end of the network. One side uses the forward propagation and the other for the attention and then it does the same with the opposite direction as the equation as follows:

$$y_1 = x_1 + Attention(x_2) \quad (6)$$

102

$$y_2 = x_2 + \text{FeedFwd}(y_1) \quad (7)$$

103 Recomputing x_1, x_2 from y_1, y_2 :

$$x_1 = y_1 - \text{Attention}(x_2) \quad (8)$$

104

$$x_2 = y_2 - \text{FeedFwd}(y_1) \quad (9)$$

105 Using two branch of the network, when it come back for the back propagation the y is
 106 computed to get the x_2 and x_2 then used with y_1 to calculate x_1 that requires no weight to
 107 store.

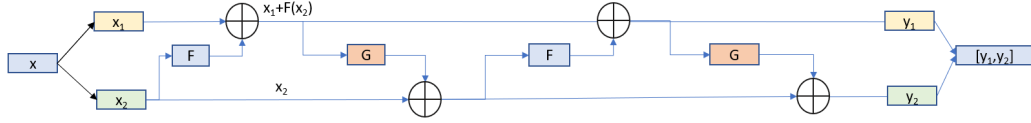


Figure 2: Reversible Residual Layer Model

108 3.4 Reversible Transformer

109 The reversible transformer (RT) is the combination of two techniques that resolves memory
 110 allocation and attention problems which relies on the transformer network. We used LSH
 111 to reduce the complexity of the MA over LS. Then the RRL has been used to use available
 112 memory efficiently. Using the Trax framework the RT has been implemented to build the
 113 chatbot with the large context window governing multiple domains and topics for the human
 114 conversation. For the RT model implemented in Trax we used the following architecture:

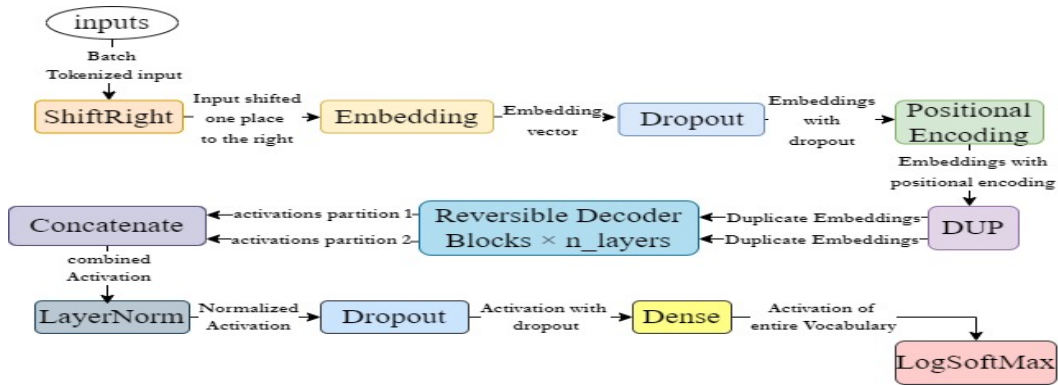


Figure 3: Reversible Transformer Model

115 The attention layer and the feed forward layer has been implemented in input on our model.
 116 The memory efficiency has been improved by using the reversible decoder block in Trax
 117 framework. The general model of the architecture is shown below:

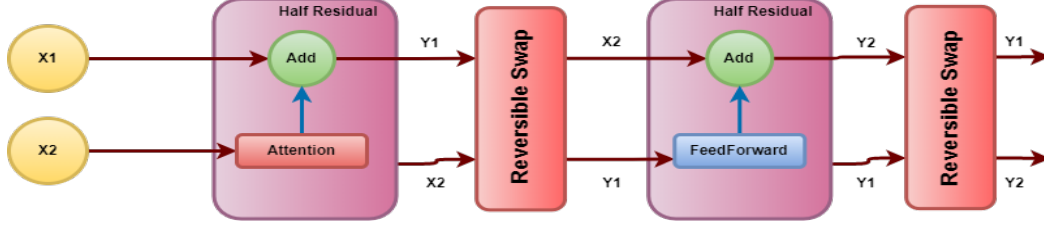


Figure 4: Reversible Decoder Block

From the model, it is visible that the initial input x_1 and x_2 implements the equation (6, 7) of the RT network. The reversible residual equation for the forward pass that states one equation is the half portion of the reversible decoder block. Before the second equation implementation it need to swap the elements to take account the stack semantics in Trax. Trax puts the x_2 on the top of the stack to feed the add block of the half residual layer. After that we swap the two outputs again before feeding to the next layer to implement the second equation and then the block is used to recompute the activation during the backward pass.

4 Implementation

4.1 Dataset

The MultiWoz 2.1 dataset is been used as the primary dataset for our model. This dataset has more than 10000 human annotated dialogues and has the spans of multiple domains and topics (12). The dataset contains “MUL” for the multi domain dialogues and have “WOZ” or “SNG” for the single domain. The dataset is downloaded in JSON format with one key value pair for the one dialogue. We extracted the conversation from the dataset by setting offset of even for the one person and the odd for the second person from the log list. The dataset contains the calls, hotel, hospital, doctor, taxi, train, police and restaurant databases to automate the entire conversation with the chatbot model and generate useful responses. For the Bangla language we created our dataset সমাচার (Shomachar) with 1000 human annotated dialogue with the same domain of the MultiWoz dataset. The dataset also formatted in JSON format and all the attributes are kept the same as the MultiWoz.

4.2 Data Processing

As the ‘Person 1’ and ‘Person 2’ acts as delimiters, the model can easily recognize the conversation and identify the next person. For processing the data for the RT input with corresponding text response from the each person we first untokenized the data. Then it is splitted into train set and the evaluation set. The both set has been tokenized with batches and we made a tuple pair for the identical values. After that, the data has been pipelined for tokenizing and batching. Then the bucket is created by its length and has the upper bound on the token length.

4.3 LSH Attention

We used Trax a neural network development platform. It uses ‘layers’ for the abstraction. For the LSH implementation and add attention mechanism Trax has classes for the attention layers. The base.layer class has *EfficientAttentionBase* that leaves many routines to be overridden by child classes. However, *use_reference_code* makes it capable to implement custom codes that limits complexity. It implements a nested loop that treats every example head.

154 4.3.1 Hash Vector

155 Trax’s hash vector has been reimplemented with modification. An array of vectors has been
 156 taken and it hashes the entries then returns the array with assigned input with hash bucket.

```

Define ← rotate_shape;
Define ← randomize_rotation;
if fastmath.backend_name() = 'jax' then
  | rotated_vectors ← np.einsum('tf, fhb -> htb', vecs, random_rotations);
else
  | Calculate random rotation and random vectors;
  | Concatenate vectors;
  bucket ← np.argmax(rotated_vecs, axis = -1).astype(np.int32);
  if mask is not None then
    | n_buckets ← n_buckets + 1;
    | buckets ← np.where(mask[None, :], buckets, n_buckets - 1);
  end
end
end

```

Algorithm 1: Hash Vector

157 4.3.2 Sorting bucket

158 The generated bucket is $n_hash * n_seq$ long and has been offset by n_hash as numbers
 159 cannot overlap. Then the bucket is sorted to group together as (hash, bucket) pair.

160 4.3.3 Chunked attention

161 After sorting Q we perform dot product for the attention mechanism. *numpy.matmul*
 162 will stack the martices of the inputs residing the last two indexes. Then we performed
 163 *softmax* on the output. However, the hash has multiple hash tables that *softmax* need to
 164 be performed separately and need to sum up (13). For the efficiency we slightly reorganized
 165 the *softmax* function as below:

$$softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (10)$$

$$logsumexp(x) = \log\left(\sum_j \exp(x_j)\right) \quad (11)$$

$$softmax(x_i) = \exp(x_i - logsumexp(x)) \quad (12)$$

```

Define ← reshapeandswap_axes;
Dot ← np.matmul(reshape, swapaxes);
Use softmax on Dot with passthrough;
np.reshapeV such that middle dimension is size of chunk;
S ← np.matmul(Dot, V);
Reshape 2D array with no change in last dimension;
Reshape logits;
O ← undo sort of V with axis = 0 and logits;
if n_hashes > 1 then
  | O ← Combine all hashes;
end
end

```

Algorithm 2: Chunked Attention

166 4.3.4 LSH with attention

167 We split some of the functionality in our routines between attend and *forward_uncatched*.

```

Q = np.matmul(X, weight_q);
V = np.matmul(X, weight_v);
if update_state then
    Define ← State, RandomHash, Bucket;
    if self._max_length_for_buckets then
        | Length = n_hash * maxlen;
    end
    if bucket.shape[0] < length then
        | Concatenate bucket;
    end
else
    Bucket ← state;
    if self._max_length_for_bucket then
        | Reshape bucket with axis=0;
    end
    SeqLen ← x.shape[0];
    Assert shape;
    implement algorithm 2;
    Use attend function form Trax 1.3.4;
end

```

Algorithm 3: LSH with attention

168 4.4 Reversible Residual Network

169 The general approach is to store the outputs of each stage for using back propagation.
170 Running the algorithm reverse using the output gets the reversible layer. We used Trax
171 reversible layers to implement the network. The implementation starts by duplicating the
172 inputs of two paths. By copying the top stack and pushing the next two copies of the stack
173 this is performed. Then it is fed to the half residual layer. This will be repeated until we
174 reach the end of the reversible serial section. Then it is concatenated.

175 4.5 Reversible Transformer Model

176 After generating reversible layers we randomized it as for the backward pass
177 when random noise is included it may return the correct layer input. Using
178 *trax.fastmath.random.uniform()* we randomized the layer first. Then we go for the train-
179 ing of our model implementing figure 3. We used Trax’s ReformerLM to perform the imple-
180 mentation.

```

Serial ←
    ShiftRight(1), Embedding_train_512, Dropout, PositionalEncoding, Dup_out2;
ReversibleSerial_in2_out2 ← ReversibleHalfResidualV2_in2_out2;
    LayerNorm;
SelfAttention;
ReversibleSerial_in2_out2 ← ReversibleHalfResidualV2_in2_out2;
    LayerNorm, Dense_2048, Dropout, FastGelu, Dense_512, Dropout;
do ← continue;
Concatenate_in2;
LayerNorm;
Dropout;
Dense_train;
LogSoftmax;

```

Algorithm 4: Reversible Transformer Model

181 For the training purpose the training loop has generated with the Train and Evaluation set.
182 Then the loop is initialized with *trax.supervised.training.Loop*. The labeled data is set to
183 *train_gen* and for loss function we used *CrossEntropyLoss*. We used the Adam optimizer

184 with the *lr* scheduler. The *n_steps_per_checkpoint* is set to 10. The evaluation set is
 185 fixed with the labeled *data = eval_gen* and the accuracy and loss is calculated.

186 Then the output has been generated from the model with the
 187 *trax.supervised.decoding.autoregressive_sample_stream* function with the *x* batch
 188 size and batch dimension array has been added. The dialogue generator function of the
 189 chatbot is given below in form of algorithm:

```

Define  $\leftarrow$  delimiter1, delimiter2;
Initialize token counter;
Result  $\leftarrow$  tokenizer;
ReformerLM.state  $\leftarrow$  model_state;
Output  $\leftarrow$  generated_model_output;
For O in output;
Append result;
Detokenize output;
if sentence.endswith(delimiter1) then
  | Sentence  $\leftarrow$  split_with_delimiter1;
  | Print_delimiter2_sentence;
end
Result.clear();
Add counter;
if counter > max_len then
  | Break
end

```

Algorithm 5: Dialogue Generator

190 5 Result

191 Our model achieved 98% accuracy while training on the evaluation set. The outcome of the
 192 model is a full functional chatbot that can reply with meaningful sentences and generate the
 193 conversation. The model can be converted in any language based on the dataset it has been
 194 feed. For illustrate we used Bangla language and the chatbot can reply in Bengali. The
 195 figure below shows the two form of output based on languages. One output is in English
 196 and another is in Bengali.

197 Person 1 : Is there a hospital nearby?
 Chatbot: Addensbrookes Hospital is located at Hills Rd,Cambridge, postcode
 CB20QQ. Do you need a particular department?
 Person 1: No, I just need the phone number, please.
 Chatbot: The phone number is 81223245151.
 Person 1: Thank you. That is all I need.
 Chatbot: Thank you for using our services. Goodbye.
 Person 1: Goodbye.

198 Person 1: আপনি একটি ট্যাক্সি বুক করতে পারেন?
 Chatbot: আমি নিশ্চিত করতে পারি। আপনি কখন চলে যেতে চান?
 Person 1: আমি 13:00 পরে যেতে হবে.
 Chatbot: আমি আপনার অনুরোধে সাহায্য করতে পেরে খুশি হব, প্রথমে আমাকে আপনার গন্তব্য
 জানতে হবে।
 Person 1: ব্র্যাক বিশ্ববিদ্যালয়
 Chatbot: বুকিং সম্পন্ন! বুক করা গাড়ির ধরন: ধূসর টয়োটা যোগাযোগের নম্বর:01726237282
 Person 1: ধন্যবাদ
 Chatbot: ধন্যবাদ বিদায়বিদায়বিদায়বিদায়বিদায়বিদায়বিদায়বিদায়

6 Conclusion

An introduction to the development of a Bangla chatbot to diagnose and solve problems is presented in this paper. It is developed a framework that supports Bangla text for enhanced machine comprehension. In terms of Bangla content, the framework uses AI to improve NLP and machine learning approaches. This model employs multi-head attention (MA), locality sensitive hashing (LSH) for the word vector, reversible residual layers (RRL) for handling long sequences (LS), and we utilized LSH to minimize the complexity of the MA over LS to run quickly and achieve a decent result. Future research should maybe concentrate on determining the best way to integrate the many techniques to chatbot development in order to maximize the effectiveness of each strategy, even the most basic.

References

- [1] Daniel Jurafsky & James H. Martin, “Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition”, Prentice Hall Inc Publications, 2nd Edition, 1999.
- [2] Nagarhalli, T. P., Vaze, V., & Rana, N. K. (2020, March). A review of current trends in the development of chatbot systems. In 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS) (pp. 706-710). IEEE.
- [3] Mnasri, M. (2019). Recent advances in conversational NLP: Towards the standardization of Chatbot building. arXiv preprint arXiv:1903.09025.
- [4] PTI, “Chabot, Virtual Assistants Can Reduce a Manager’s Workload by 70% by 2024: Gartner”, published 23rd Jan, 2020. Available at <https://www.news18.com/news/tech/chatbots-virtual-assistants-can-reduce-a-managers-workload-by-70-by-2024-gartner-2470345.html>.
- [5] Alhassan, N. A., Saad Albarrak, A., Bhatia, S., & Agarwal, P. (2022). A Novel Framework for Arabic Dialect Chatbot Using Machine Learning. Computational Intelligence and Neuroscience, 2022.
- [6] Rahman Joy, M., Akash, S., Nasib, M., & Hasan, K. M. (2021). An Intelligent Bangla Conversational Agent: TUNI. In Proceedings of International Joint Conference on Advances in Computational Intelligence (pp. 417-430). Springer, Singapore.
- [7] Rahman, M. M., Amin, R., Liton, M. N. K., & Hossain, N. (2019, December). Disha: An implementation of machine learning based Bangla healthcare Chatbot. In 2019 22nd International Conference on Computer and Information Technology (ICCIT) (pp. 1-6). IEEE.
- [8] Elcholiqi, A., & Musdholifah, A. (2020). Chatbot in bahasa Indonesia using NLP to provide banking information. IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 14(1), 91-102.
- [9] Fabian, R., & Alexandru-Nicolae, M. (2009, September). Natural language processing implementation on Romanian ChatBot. In WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering (No. 5). WSEAS.
- [10] Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The efficient transformer. arXiv preprint arXiv:2001.04451.
- [11] Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The efficient transformer. arXiv preprint arXiv:2001.04451.
- [12] Budzianowski, P., Wen, T. H., Tseng, B. H., Casanueva, I., Ultes, S., Ramadan, O., & Gašić, M. (2018). MultiWOZ—A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. arXiv preprint arXiv:1810.00278.
- [13] Jurafsky, D., & Martin, J. H. (2018). Speech and language processing (draft). preparation [cited 2020 June 1] Available from: <https://web.stanford.edu/~jurafsky/slp3>.