Transforming Second Language Acquisition Modeling

Nathan Dalal
Department of Computer Science
Stanford University
nathanhd@stanford.edu

Nidhi Manoj
Department of Computer Science
Stanford University
nmanoj@stanford.edu

Abstract

As more student data becomes available, deep learning techniques can enable stronger analysis of tasks such as language acquisition and allow for more personalized online learning. Duolingo's Second Language Acquisition Modeling (SLAM) task provides a large corpus of beginner student data to trace how students learn a new language through three different exercises. Doing both language modeling for each exercise and simultaneously using information on student progression over time, SLAM is both a language modeling and cognitive modeling task. We propose to apply various recurrent architectures to SLAM, at the instance, exercise, and user levels.

Our main models included a logistic regression model and an exercise level LSTM model. By introducing word embeddings in our models, both our models outperform the Duolingo baseline. Increases in performance from adding a one-hot user encoding shows evidence that learning about different users helps performance. Sentence structure of the exercise and deeper networks clearly matters as well, given the performance improvements moving from Logistic Regression (which was on the instance level) to an exercise level LSTM. Our best model, the exercise level LSTM model, would currently sit at 7th place on the public worldwide leaderboard of all submissions to the English language learning track, with an AUC of 0.840. Our performance is comparable to the 4th-6th places on the leaderboard which have AUC between 0.840 and 0.850. Additionally, we have only hyperparameter tuned the learning rate, and since we have yet to overfit the model we have avoidable bias, indicating that deeper networks and hyperparameter tuning can get us past the current state-of-the-art English track AUC of 0.861. We are continuing to work on the project and will release a paper and results once we beat the state-of-the-art.

1 Introduction

1.1 What is SLAM?

In the modern, well-connected world where individuals want to be able to communicate with others, the need to improve our instruction of Second Language Acquisition (SLA) grows. Online language learning through platforms such as Duolingo allow learners around the world to learn a target language (L2) from their source language (L1). The Second Language Acquisition Modeling (SLAM) dataset provides us with rich student learning data to understand how people learn L2.

Duolingo's Shared Task on Second Language Acquisition Modeling (Settles et al. [2018]) provides publicly available data to study SLA. The associated dataset and modeling task was released as

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

a timed challenge. The paper referenced above describes the SLAM dataset as well as the top performing submissions in a SLA task. A link describing the challenge in detail can be found at `http://sharedtask.duolingo.com/`.

We choose this paper because the SLAM dataset is the largest dataset for language acquisition available and the prediction task (which we will discuss further) lies within the nascent and growing intersection of language acquisition and machine learning. Working on SLAM is an opportunity to apply new NLP models to educational applications with worldwide impact. Furthermore, SLA modeling is part of a larger movement to understand and personalize student learning. A lot of previous work has been done about acquisition of skills such as math, but language and SLA specifically, are harder to model and understand. Additionally, access to a large dataset creates a great opportunity to combine natural language modeling with recent advancements in deep learning.

1.2 Related Work

As mentioned, the top submissions of models for the SLAM prediction task were summarized in a summary paper written by Duolingo (Settles et al. [2018]). Models that attempted the SLAM task are evaluated using two metrics: AUC and F1. The primary measurement, AUC, measures the area under the ROC curve, measuring how good the model is at separating classes. The ROC curve plots the true positive rate against the false positive rate. F1, the harmonic mean of precision and accuracy, is reported as a secondary metric and is fixed at 0.5 for evaluation reporting.

Duolingo released SLAM as a challenge with a logistic regression baseline (which had AUC $.774$ and F1 0.190 for the English track), and submissions to the challenge improved on the baseline. Thirteen teams submitted results for all language tracks. The best teams used RNNs and Gradient Boosting Decision Trees to achieve their results. The team achieving the best results was from SanaLabs (Nilsson et al. [2018]) and used an ensemble of both model types to achieve the highest AUC across all three language tracks (the highest of which was $.861$ which was in English).

Analyzing across submissions, token-level features and student-level features influenced predictions the most, while linguistic and grammatical features were the least important. Words, users, and student response time were critical predictive features in most team submissions, and there was not much variation in difficulty across different language tracks.

In particular, the work of Xu et al. [2018] (also an AUC of $.861$ on the English track, tying Nilsson et al. [2018]) used 4 separate RNN encoders and 1 RNN decoder. The encoders stored information on token context, linguistic information, user data, and exercise format. They found that token context and user data provided the most use, while linguistic information did not help much. They did not use much feature engineering, a similar inference from those at the top of the leader board. This shows that the automated feature engineering of neural models is quite robust for the SLAM task, a sentiment that Duolingo's summary paper agrees with. Since, previous work shows that engineering custom features do not significantly increase results, we will also use the token and student level features for our modeling and prioritize experimenting with different model architectures over attempting to engineer new features.

Duolingo constructed an oracle from all team submissions, outputting the team prediction with the lowest error for each token. The oracle had > 0.993 AUC and > 0.884 F1 which indicates that there is still a large room for improvement on the SLAM task and a possibility to beat the current state-of-the-art model by SanaLabs.

1.3 Our Contributions

We quantitatively show that the use of transfer learning via word embeddings, explicitly modeling users with a one hot encoding, learning across words in an exercise, and running deeper models all improve results.

Our results of 0.730 AUC on logistic regression without student modeling indicate that not adding the user as an explicit one-hot encoded feature decreases model performance. When we added in user information with word embeddings, we reached an AUC of 0.781, surpassing the Duolingo baseline with no hyperparameter tuning on any attribute besides a learning rate of 0.01. This indicates that word meanings in context can effectively model student understanding better than just a one-hot encoding of words and, in fact, correspondingly led to a large gain in performance.

Moving to our LSTM architecture that loops over words in one exercise, we got results as high as 0.840 AUC, also only tuning the learning rate. Our model got better as we added more neurons (deeper model with more layers) and decreased the learning rate. Additionally, training and validation losses remained almost the same, even as the model got deeper. We can see that modeling the sentence as a sequence and adding more latent variables helps the model learn.

We also find that the training and validation distributions are remarkably similar, thanks to the lack overfitting from relatively deep models. This means that in the future we can add complexity and expect our results to get even better. We will continue to pursue our deep learning approach after the course to attempt to reach and ultimately surpass the state-of-the-art.

2 Dataset and Task

2.1 Data Summary

For our project, we will work on the Duolingo's Second Language Acquisition Modeling (SLAM) (Settles et al. [2018]) dataset and prediction task. The dataset comprises of exercise data from 6.4K students during the first 30 days of learning a language on Duolingo.

Exercises either ask students (1) to translate from the L1 language to L2 in free form, (2) to translate from L1 to L2 using a bank of words, or (3) to transcribe dictation in L2. Each exercise contains a list of L2 words (tokens) and each word is paired with a binary label for whether the student answered correctly for that given word. The dataset contains three language tracks of students learning English, Spanish, and French, with a total of 7.1M labeled tokens. We focused on the English track for now.

The data contains many student exercise sessions with student metadata information, such as how many days of experience they have with the L2 language, which platform they did the exercise on (web, android, iOS), and how long it took them to answer the exercise. Each exercise contains a list of instances, each corresponding to an L2 token (word) and corresponding linguistic information about the word in context. Each token has a corresponding binary label for whether the student answered (translated or transcribed etc) that word correctly in the exercise. This format is seen in an example of the data in Figure 1.

Figure 1: Data format organized into exercise sessions with one word on each line, accompanied with features for each word. Settles et al. [2018]

```
# user:D2inSf5+  countries:MX  days:1.793  client:web  session:lesson  format:reverse_translate  time:16
8rgJEAPw1001  She          PRON    Case=Nom|Gender=Fem|Number=Sing|Person=3|PronType=Prs|fPOS=PRON++PRP    nsubj       4  0
8rgJEAPw1002  is           VERB    Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|fPOS=VERB++VBZ   cop         4  0
8rgJEAPw1003  my           PRON    Number=Sing|Person=1|Poss=Yes|PronType=Prs|fPOS=PRON++PRP$             nmod:poss   4  1
8rgJEAPw1004  mother       NOUN    Degree=Pos|fPOS=ADJ++JJ                                               ROOT        0  1
8rgJEAPw1005  and          CONJ    fPOS=CONJ++CC                                                         cc          4  0
8rgJEAPw1006  he           PRON    Case=Nom|Gender=Masc|Number=Sing|Person=3|PronType=Prs|fPOS=PRON++PRP  nsubj       9  0
8rgJEAPw1007  is           VERB    Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|fPOS=VERB++VBZ   cop         9  0
8rgJEAPw1008  my           PRON    Number=Sing|Person=1|Poss=Yes|PronType=Prs|fPOS=PRON++PRP$             nmod:poss   9  1
8rgJEAPw1009  father       NOUN    Number=Sing|fPOS=NOUN++NN                                             conj        4  1

# user:D2inSf5+  countries:MX  days:2.689  client:web  session:practice  format:reverse_translate  time:6
oMGsnnH/0101  When         ADV     PronType=Int|fPOS=ADV++WRB                                            advmod      4  1
oMGsnnH/0102  can          AUX     VerbForm=Fin|fPOS=AUX++MD                                             aux         4  0
oMGsnnH/0103  I            PRON    Case=Nom|Number=Sing|Person=1|PronType=Prs|fPOS=PRON++PRP              nsubj       4  1
oMGsnnH/0104  help         VERB    VerbForm=Inf|fPOS=VERB++VB                                            ROOT        0  0
```

2.2 Task Description

For each instance (word token), we would like to predict whether the student got that token right or wrong. An example of the input and output is seen in Figure 2. This is a binary classification task, and can be passed in as input into models at the instance level (one word at a time), at the exercise level (one exercise or list of words at a time), or at the user level (one student at a time). We tried a model for each of these levels: Logistic Regression Baseline, ExerciseLSTM, and UserLSTM respectively.

Figure 2: We see a typo on 'When' and a missing pronoun 'I' in the input phrase above. Settles et al. [2018]

learner:	wen	can		help	
reference:	when	can	I	help	?
label:	✗	✓	✗	✓	

3 Approach

3.1 Data Processing and Feature Extraction

Duolingo provided us with very basic skeleton code for reading in the file and placing the results in a list of word-level objects. We significantly modified this code to produce a list of objects on an exercise level and built a feature extraction system to convert this exercise object into numerical Tensor features (Pytorch) that can be used in a deep model. We wrote all this data loading by ourselves as well as all the models from scratch. We also did extra data exploration to better understand the words, sequences, users in the dataset better: for example, we found that the maximum number of words in an exercise was 14 so we padded sequences to 20 words to allow some breathing room.

3.1.1 Exercise-Level Features

Here are the exercise-level features we encoded.

- Country: one-hot encoding of 40 possible countries (length 40)
- Days: Days of experience with this language on Duolingo, encoded as one number.
- Client: one-hot encoding of 3 possible platforms for Duolingo (web, ios, android)
- Session: one-hot encoding of 3 possible exercise settings (lesson, practice, test)
- Format: one-hot encoding of 3 possible session types (reverse_translate, reverse_tap, listen)
- Time: seconds it took for student to submit the exercise, which we encode as one number.
- User: base-64 unique user id of length 8 used as a one-hot encoding of 2593 possible users.
  - We struggled with encoding the user properly, until upon the analysis of the data we saw that all users that appeared in the train, dev, and test set were the same and so we decided to one-hot encode the user into a user embedding.

Here is what we omitted from feature encoding.

- Prompt: The prompt in the user's L1 is given to us, another sequence to model. We have not gotten around to modeling this sequence, and we will do so if we have more time.

3.1.2 Word-Level Features

Here is what we encoded for each instance.

- Part of Speech: One hot encoding of 17 possible different parts of speech
- Dependency Label: A number indicating other words it depends on in the sentence. Tokens can have many dependency labels, so this is many-hot encoded (length 37)
- Dependency Head Edge: a link to another index, encoded as one number feature
- Token: Loaded as an embedding of size 300, explained in the section below

We omitted morphological features, long structured strings of all grammatical morphological features. We did not find a structured way to encode this, and since other papers that worked on SLAM in related work found these features to be insignificant, we omitted them.

3.2 Token Embedding

We wanted to use word embeddings (Mikolov et al. [2013]) to do transfer learning from larger language corpora. We leverage MUSE, developed by Facebook, which supports word embeddings for many languages Lample et al. [2017]. We took the English, Spanish, and French word embeddings from here to use in our models (embedding size 300).

Lample et al. [2017] have also trained bilingual embeddings that are learned by attempting to translate from one language to another (rather than general embeddings for a language), so as future work we will leverage these more applicable embeddings to our models.

In our usage of MUSE embeddings, we converted each token to an index that allows us to look up the corresponding language embedding for that token. From scratch, we wrote the code to parse the embeddings, to add padding and unknown tokens, and to establish the use of word embeddings throughout all of our deep models. For all modeling, we train into our embeddings so that as the model learns, the embeddings for specific words are fine-tuned for the SLAM task.

3.3 Data Loading

Once we converted all the data into numerical data-label pairs, we created data loaders that support both CPU and GPU training from scratch. We built data loaders to train on both individual instance-level as well as exercise-level, so that we can train on an instance-level Logistic Regression model and a exercise-level LSTM. We wrote the data loader code ourselves and just that was a significant 30+ hour endeavor. These data loaders allow us to train on different batch sizes natively with PyTorch. We also padded our word sequences and exercises ourselves to handle variable sequence lengths.

For running models, each of our models specify a method to transform the data, and a method to execute a forward pass. For our LSTM models and other sequence models going forward, we also built list padding and masking methods to train on fixed length sequences, and adapted our evaluation methods to support this. The data loading and processing was a lot of work for this custom project but this eases future work and can be used and generalized easily when training future sequential models.

3.4 Implemented Models

3.4.1 Training Method (Loss Function)

Since all models output logits, we optimize with a Sigmoid activation into a Binary Cross Entropy loss function. For all sequence models, since we train on fixed length sequences, some masking is involved to optimize the loss correctly, since we must average over the true number of words per exercises (in the exercise LSTM case) and the true number of exercises per group for a user (in the user LSTM case). We implemented this masking to ensure the loss was calculated on only valid inputs. This was non-trivial and was around a 10 hour endeavour with debugging.

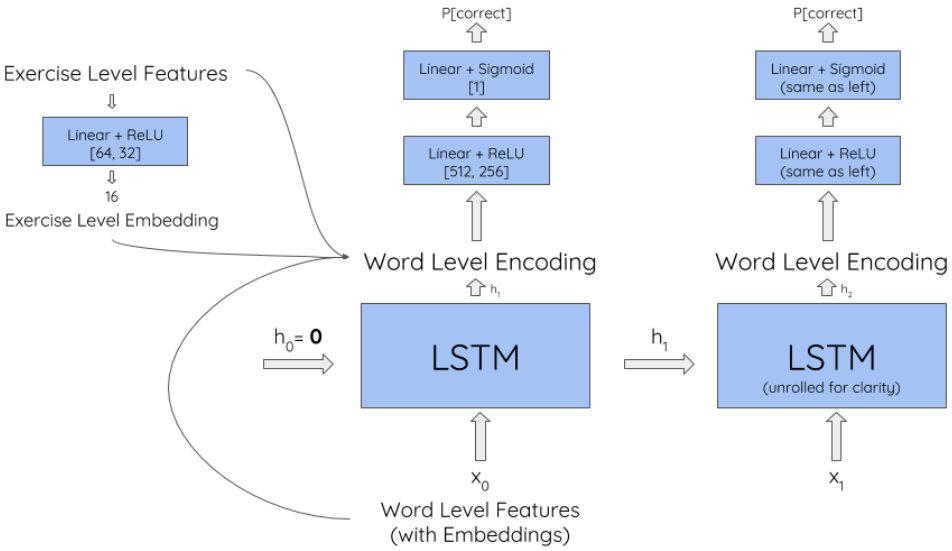
3.4.2 Baseline

Logistic Regression operates with one token as one training example. It takes concatenated word-level and exercise-level features as input and then outputs one logit number. The output is a binary prediction if the user will get the word correct or not. It is a zero-layer neural network with a sigmoid activation (used in the loss). A logistic regression is a standard baseline for binary classification tasks but converting everything into PyTorch etc took time.

3.4.3 Exercise LSTM

The LSTM operates with one exercise (containing a sequence of words) as one training example. The LSTM passes the exercise-level features through a neural network to encode them. The first hidden and cell states are vectors of zeros. An LSTM is applied through each set of word-level features for the exercise. Then, a combination of the hidden states from each timestep, the original word embeddings and word level features, exercise level features, and a deep encoding of the exercise features, are passed through a neural network layer to produce one binary output for each hidden state. We integrate skip connections to the final neural network for the original word level features and exercise level features. We also tried appending on the user 1-hot encoding to the features in various ways and found that we had the best performance when it was concatenated onto the exercise level features (and skip connections were used) which presumably allowed the model to retain information of the user throughout and thus learn better.

Figure 3: Architecture for the exercise level LSTM.



3.4.4 User LSTM

We also built and attempted to train a User LSTM, that operates on groups of 25 exercises of the user at a time for each user. Our motivation for this approach was to model forgetting over time, and to have our model really learn about a user's progress over time. Our approach was to send the last valid hidden and cell states of the previous exercise through a neural network, and then use them as the next initial hidden and cell states. Batches of 25 exercises give us a chance to model a group of exercises. Like all the otehr models, we wrote this from scratch. Unfortunately the loss was unstable and the model did not learn at all (AUC near 0.5), with results remaining the same regardless of learning rate.

We do think modeling a user over time is important, and we implicitly do it in the exercise LSTM with the time feature (which other groups in related work also noted to be a significant feature). We will likely revisit this model as a stacked LSTM architecture in the future, but for now we are happy with the strong results from our deep exercise LSTM model.

4 Experiments

4.1 Evaluation Method

As explained in the Related Work section, models that attempted the SLAM task are evaluated using two metrics: AUC (area under the ROC curve) and F1 (the harmonic mean of precision and accuracy). We used AUC (area under the ROC curve) as the primary evaluation and F1 as a secondary metric.

4.2 Experimental details

For all models, we use Xavier initialization and Adam optimizer. Our learning rate was reduced on plateau and the weight decay is set to 0.00001. Larger batch sizes improved all the model's speed and performance so we tried various batch sizes that would fit on the machine. Logistic Regression with user embedding took around 6 minute per epoch to train. LSTM took around 7 minutes to train per epoch. We tuned our models for various initial learning rates. Further model configurations are noted in the results table below.

4.3 Results

Table: Performance of models. Our models are noted with an asterix.

Model	AUC	F1	Accuracy	LR	Batch	Epochs
State-of-the-art Sana Labs	0.861	0.561	–	–	32	–
∗Our ExerciseLSTM	0.840	0.423	0.880	0.001	256	12
∗Our User Logistic Regression	0.781	0.203	0.863	0.01	4096	9
Duolingo Baseline	0.774	0.190	–	–	–	–
∗Our Non-User Logistic Regression	0.730	0.111	0.858	0.01	4096	9

Our exercise level LSTM results are good for 7th place on the leaderboard from the SLAM challenge. In fact, our results are more comparable to places 4-6 that have performance between 0.840 and 0.850, rather than places 7 onward, all with AUC below 0.830, indicating that our performance among researchers who submitted to the competition is better than average.

We found that the Logistic Regression without any user encoding achieved AUC 0.730 and F1 of 0.111 at epoch 9. Employing user 1-hot encodings then boosted the Logistic regression performance to AUC 0.781 and F1 of 0.203 at epoch 9. This surpassed the 0.774 baseline English AUC achieved by the SLAM baseline model.

The Exercise level LSTM model, without hyperparameter tuning on any attribute other than initial learning rate, achieved AUC of 0.861 and F1 of 0.561. We trained the LSTM with a dropout probability of 0.1.

Note that the best submitted AUC performance on the English track is .861, which is the current state-of-the-art, by Nilsson et al. [2018].

4.4 Analysis

Our results of the initial logistic regression model without user modeling (0.730 AUC) indicate that not adding the user as an explicit one-hot encoded feature decreases model performance. When we added in user information with word embeddings, we reached an AUC of 0.781, surpassing the Duolingo baseline with no hyperparameter tuning on any attribute besides initial learning rate. This indicates that the user is a key attribute in the task. Additionally, we see that word meanings in context (using MUSE word embeddings) can effectively model student understanding better than just a one-hot encoding of words. So we see that transfer learning via word embeddings and explicitly modeling users even with just a with a one hot encoding leads to a large gain in performance. We presumed that user information would be helpful after reading related work but we did not anticipate that the performance gain would be so significant.

We also hypothesized that learning across words in an exercise (sequence of tokens) will be beneficial and thus moved from an instance level model to the exercise level model. This, in fact, helped improve results. Moving to our LSTM architecture that loops over words in one exercise, led to results as high as 0.840 AUC, again only tuning the initial learning rate. We can probably get better results with further hyperparamter tuning on weight decay and and experimenting regularization and dropout. We also predicted that using deeper models would lead to better performance. This proved true because our model got better as we added more neurons (more layers) and decreased the learning rate. We can see that modeling the sentence as a sequence and adding more latent variables helps the model learn.

Furthermore, we also noticed that the training and validation distributions are similar, thanks to the lack overfitting from relatively deep models. The losses for both decreased as desired and this means that in the future we can add complexity and expect our results to get even better.

We analyzed our results a lot and found some trends. For example, the model tends to perform better on shorter sequences than longer sequences. For example "From September to December" was predicted to be "1 1 0 1" and the true labels were in fact "1 1 0 1" with the user getting all words correct except the "to". Meanwhile, longer sequences such as "November is a month of the year was predicted to be "1 0 0 1 0 0 1" but the true labels were "1 1 0 1 1 0 1" predicting user performance on almost half the words incorrectly. Other trend we noticed were that more common words in English had better predictive performance. This could be because gradients are not carrying through for longer sequences and attention or a deeper model is needed. Lastly, we noticed that the model tended to predict negatives (incorrect translation) more often than positives (correct translation). This is

because there is class imbalance with more negatives in the dataset (see the confusion matrix below). Though it had decent performance for both positives and negatives, performance on negatives was better (predicting negatives correctly 90% of the time).

Figure 4: Confusion Matrix of LSTM results.

Confusion Matrix for LSTM results		
	Actual Positive	**Actual Negative**
Predicted Positive	True positives = 16999	False positives = 8405
Predicted Negative	False negatives = 38359	True negatives = 323611
	Total actual positives: 55358	Total actual negatives: 332016

5 Conclusion and Future Work

We have very good performance with our ExerciseLSTM on this task and would be 7th on the worldwide public leaderboard for performance on the SLAM task with an AUC of 0.840. We will continue working on the model after the course to get better results and try to beat the current state-of-the-art model on this task (AUC 0.861). We are both very interested in the task and the intersection of language and education.

In future work, we will try hyperparameter tuning for weight decay and other attributes and experiment with dropoout and regularization to improve both LSTM and Logistic Regression. We want to add bidirectionality to the LSTM. We also plan to try making the model deeper since we are not overfitting. Lastly, we plan to debug the UserLSTM because we believe that using user progression through exercises over time can boost performance. The model has vanishing gradients right now because each user has many excercises so we will try to tweak that architecture.

Overall we thoroughlly enjoyed working on this task, which was non-trivial but very rewarding. We were very pleased to have achieved such great performance and are excited to keep working on the project after the course. We hope to keep working on the project and release a paper and results on arxiv soon.

6 Acknowledgements

This is a custom project and Michael Hahn was our CS224N mentor - we thank him for his support and feedback.We would also like to thank Arjun Manoj, an external contributor on the project.

References

B. Settles, C. Brust, E. Gustafson, M. Hagiwara, and N. Madnani. Second language acquisition modeling. In *Proceedings of the NAACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*. ACL, 2018.

Susanna Nilsson, Anton Osika, Andrii Sydorchuk, Faruk Sahin, and Anders Huss. Second language acquisition modeling: An ensemble approach. *CoRR*, abs/1806.04525, 2018. URL `http://arxiv.org/abs/1806.04525`.

Shuyao Xu, Jin Chen, and Long Qin. Cluf: a neural model for second language acquisition modeling. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 374–380, 2018.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*, 2017.