Towards Interactive Language Modeling

Maartje ter Hoeve* University of Amsterdam m.a.terhoeve@uva.nl
Evgeny Kharitonov Meta AI Labs kharitonov@fb.com
Dieuwke Hupkes Meta AI Labs dieuwkehupkes@fb.com
Emmanuel Dupoux Meta AI Labs dpx@fb.com

Abstract

Interaction between caregivers and children plays a critical role in human language acquisition and development. Given this observation, it is remarkable that explicit interaction plays little to no role in artificial language modeling—which also targets the acquisition of human language, yet by artificial models. Moreover, an interactive approach to language modeling has the potential to make language models substantially more versatile and to considerably impact downstream applications. Motivated by these considerations, we pioneer the space of interactive language modeling. First we present a road map in which we detail the steps that need to be taken towards interactive language modeling. We then lead by example and take the first steps on this road map, showing the initial feasibility of our approach. As such, this work aims to be the start of a larger research agenda on interactive language modeling.

1 Introduction

Interaction between children and more advanced language interlocutors (such as caregivers) plays an important role in many theories and studies on human language acquisition (e.g., Bruner, 1985; Clark, 2018). For example, although culturally dependent (Shneidman and Goldin-Meadow, 2012) and with the precise effects still up for discussion (Cristia et al., 2019), caregivers can communicate with their children in Child Directed Speech. In turn, children can for example experiment with the meaning of words, to elicit a response from their caregivers (Gillis and Schaerlaekens, 2000).

Despite the importance of interaction in human language acquisition, interaction plays little to no role in artificial language modeling. This is remarkable, as language modeling also has the objective to learn human language, albeit with artificial models.

*Work done while interning at Meta AI Labs.

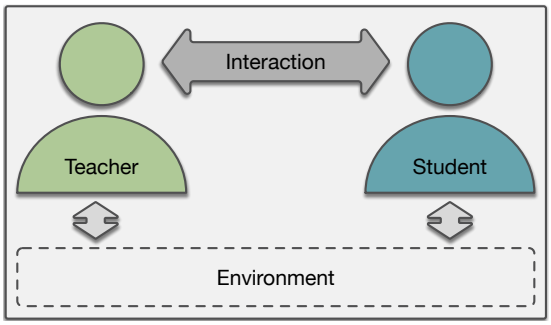


Figure 1: Teacher-Student setup for interactive language modeling.

Instead, current state-of-the-art language models (LMs) take large amounts of text as input, and are tasked to predict the next or masked words (e.g., Devlin et al., 2018; Brown et al., 2020). The learning signal only comes from a cross-entropy loss that indicates whether a prediction was correct. Although this setup has shown to be effective, from the perspective of human language acquisition it appears very unnatural. This gives rise to the motivation to investigate other, more natural approaches to language modeling, such as the interactive perspective that we propose in this paper.

Specifically, we structure our proposal according to a teacher-student setup. Figure 1 depicts a high level overview. In this setup we distinguish four main parts: *the teacher*, whose role is inspired by the caregiver in the human language acquisition, *the student*, who resembles the child, *the interaction* between the teacher and the student and *the environment* that they both share (such as the language that needs to be learned by the student). We motivate and detail our setup further in Section 3.

An interactive approach to language modeling is not only interesting from the perspective of human language acquisition. Explicitly allowing for interaction also has the potential to make language modeling more efficient and versatile. For example, a teacher can adapt its input to a student based on the specific feedback signals it receives from the

student, and a teacher that is fluent in one domain can teach the specifics of that domain to a student trained on another domain, and vice versa. Moreover, an interactive approach to language modeling has the potential to impact downstream applications, for example for foreign language teaching apps where a student can be replaced by a human.

In this paper we pioneer the space of interactive language modeling. Specifically, we contribute:

C1 We define the objective of interactive language modeling;

C2 We present a road map that details the steps that need to be taken towards this objective;

C3 We take the first steps on this road map, which show the initial feasibility of our approach.

By doing so we aim to start a larger research agenda on interactive language modeling.

2 Related Work

Over the years many different types of learning strategies have been proposed for artificial modeling. Below we describe a number of them that are particularly related to the current work.

2.1 Interactive Language Learning in NLP

Recently, a number of studies have focused on interactive language learning. Stein et al. (2021) learn logical semantic representations in an interactive way. Nikolaus and Fourtassi (2021) propose a proof of concept to model perception and production based learning of semantic knowledge acquisition in children. Kiseleva et al. (2021) take an interactive approach to language *understanding* in a recent NeurIPS challenge. To the best of our knowledge, none of the existing works have focused specifically on language modeling.

2.2 Curriculum Learning

Curriculum Learning (CL) (Bengio et al., 2009) is an approach to learning in which data samples are presented in a meaningful order—typically in order of complexity—motivated by the idea that humans learn in a similar way. Bengio et al. show the effectiveness of CL on a number of tasks, among which a classical approach to language modeling. More recently, a number of studies have shown the effectiveness of CL for (fine-tuning) LMs (Xu et al., 2020; Zhang et al., 2021), although other studies have shown that not all intuitive curricula are also effective (Liu et al., 2019). Matiisen et al. (2019) propose a teacher-student framework for automatic CL for the addition of decimal numbers and navigation in Minecraft.

2.3 Active Learning

Active Learning (AL) (Cohn et al., 1996) is an approach in which a learner (the model to be trained) actively selects which data it can most effectively be trained on. That is, where CL is often more associated with choosing a teaching strategy, AL is rather focused on the student side. AL is often used to efficiently label data in a low resource setting (e.g., Reichart et al., 2008; Dor et al., 2020).

2.4 Continual Learning

In Continual Learning, or life-long learning, the aim is to train a model in an online fashion, i.e., on a continuous stream of data, whilst avoiding *catastrophic forgetting* (McCloskey and Cohen, 1989; French, 1999). This makes models versatile to an ever changing world. Some recent work has focused on types of Continual Learning for large LMs (e.g., Lazaridou et al., 2021; Jin et al., 2021). We envision interactive language modeling to play an important role in life-long learning in the future.

3 A Road Map towards Interactive Language Modeling

In this section we present a general road map towards interactive language modeling.

Our objective is to build an automated teacher-student loop for language modeling that attains good performance in the student for a fixed (low) number of bits transmitted in the interactions.

We propose a teacher-student loop as this format closely resembles caregiver-child interactions. In Section 1 and Figure 1 we already introduced a high level overview of this setup and its four main components: (1) *the teacher*, (2) *the student*, (3) *the interaction* and (4) *the environment*. Generally, in this setup teachers transmit language data to their students, according to a certain budget ("a (low) fixed number of bits"). Having this budget forces the teacher to actively choose a learning strategy, as just sending all data that is available to the teacher would not be allowed. Students have the objective to learn the language and they send a signal back that informs their teacher of their performance, e.g., a score on an exam. This interaction takes place in an environment, e.g., a common language.

In Table 1 we present the road map that we envision towards interactive language modeling. This

road map works as follows. For each of the four aforementioned components we detail steps that need to be taken. We also add a fifth component: the evaluation of the setup. Each component has different aspects (bold-faced in Table 1). For example, for the *teacher* we can focus on how it can access the data that it can transmit to the student, which we call "ways of speaking" in Table 1. Another aspect of the teacher side focuses on what we call the "degree of awareness", which entails different ways in which the teacher can remember different aspects of the teaching loop. In a similar fashion we fill in the remaining components in the table. We focus on text as a single modality and acknowledge grounded interactive language modeling as an interesting future research direction.

On our road map there are multiple ways to reach the destination. For example, one can focus on taking a few steps for each of the components, or to take many steps for only one or a few of the components. Moreover, although mostly structured in increasing degree of complexity, this does not always hold for all individual steps in the table. For example, zooming in on the "degrees of awareness" for the teacher again, one could imagine an example where a teacher does not have an explicit memory buffer of what it sent to the student before, but does have an explicit way of remembering what the student's fine-grained capabilities are, as well as the other way around.

In the remainder of this work we take the first steps on the road map. We focus on the teacher side, i.e., learning the correct didactic approach.

4 Taking the First Steps on the Road Map

Figure 2 shows how we adapt the general setup from Figure 1 to take the first steps on the road map. Here we describe each modification per component: *the teacher*, *the student*, *the interaction*, *the environment* and *the exam* that the student takes.

4.1 The Teacher

In this work we focus on the teacher side. The role of the teacher is to transmit language data that will optimally help the student to learn the language. Figure 2 shows that we train the teacher to do this in a number of time steps. At each of these steps a teacher samples data from a larger language data set according to a fixed budget. We discuss the specifics of the sampling function below. To reduce the variance in the teacher's learning process

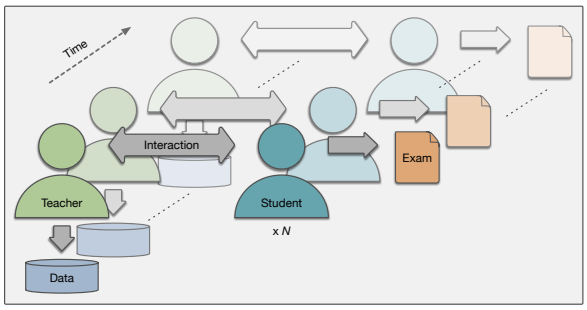


Figure 2: Teacher-student loop as used in this work.

we repeat this process for multiple students, i.e., a teacher selects N "lessons" for N students. Due to the stochasticity of the sampling process, each student has the potential to be trained on a slightly different part of the data. Because we use a multiprocessing setup we can train multiple students on a single GPU. Hence, using multiple students does not drastically increase the computational cost.

4.1.1 Knowing the Language

The teacher is modeled as a native speaker of the language that it needs to teach. We represent the teacher's language understanding with a pretrained causal Transformer LM (Vaswani et al., 2017). We pretrain this model on a *different* subset of the data than the teacher can select from for the students, and thus we ensure that we measure whether a teacher can teach a language as a whole, and not only a particular subset that it was trained on itself.

4.1.2 Selecting the Data

We use REINFORCE (Williams, 1992) with entropy regularization (Mnih et al., 2016) to learn the teacher's didactic approach.[1] We want to optimize the teacher's policy such that it learns to select the optimal data to train the student on, given a predefined budget. The policy is a one-layer feed forward neural network, that outputs a score for each sentence, i.e., the teacher's policy network takes a sentence embedding as input, based on the pretrained Transformer LM that we use to represent the teacher's language understanding. An action is modeled as selecting k sentences from the larger data set, where k is a predefined teacher budget. We use the GumbelTopK trick (Vieira, 2014; Kool et al., 2019) to sample k sentences without replacement, based on the teacher policy's output scores. We compute the log probabilities (needed to com-

[1] We also experimented with gradient-free optimization approaches such as the ones implemented in Nevergrad (Rapin and Teytaud, 2018), but found REINFORCE to be more flexible in our case and therefore a better fit for our needs.

<table>
<tr><th>Teacher</th><th>Student</th></tr>
<tr><td>Ways of speaking
• Select data from bin;
• Generate data with own language model.
Degrees of awareness
• (No) memory buffer of what has been sent to the student and being able to act on it (see Interaction cell);
• (No) explicit way of remembering what the student's fine-grained capabilities are and being able to act on it (see Interaction cell).</td><td>Ways of speaking
• Generate language data in a standard LM fashion;
• Actively experiment with language generation to elicit direct feedback from the teacher (see also Interaction cell).
Degrees of using the teacher data
• Use all data received from the teacher;
• Actively select data that is useful;
• Actively know when to stop training (for example to avoid overfitting).</td></tr>
<tr><th>Interaction</th><th>Environment</th></tr>
<tr><td>Teacher side
• Send all data at once;
• Send data in batches, based on student feedback (see below). Batches can be as small as single utterances, after which the student sends an utterance back, like in real human-to-human interaction (see below);
• Send (mid-term) exams.
Student side
• Send a single average exam score back to the teacher;
• Send a fine-grained exam score back, e.g.,
– score per item on the exam set;
– (average) scores of different components (tasks) of the exam(s)
• Ask for feedback, for example by actively experiment with language generation for the teacher to judge ('generate own exam').</td><td>Language
• Artificial languages, in increasing level of difficulty in terms of complexity, e.g.,
– random language;
– different types of structures;
– different vocabulary sizes;
• Subset of human language, e.g., in terms of
– semantics (e.g., different domains)
– syntax (e.g., different grammatical structures)
– pragmatics
• Unrestricted human language.
Task
• Teacher: Learn to select or generate the optimal data such that the student performs well on the exam set (see cell below);
• Teacher: Learn to adapt to different types of students, e.g.,
– architectural differences
– different prior knowledge (be aware of catastrophic forgetting in neural networks)
• Student: Learn to adapt to different types of teachers (didactic strategies).</td></tr>
<tr><th colspan="2">Evaluation / Exam</th></tr>
<tr><td colspan="2">Teacher
• Accuracy in selecting the optimal teaching protocol
Student (Exam / Feedback for teacher)
• General performance, measured in perplexity;
• Performance on specific tasks, such as
– Subset of the data known to the teacher (e.g., specific domain or (grammatical) structure)
– BLIMP (Warstadt et al., 2020);
– BIG-Bench (https://github.com/google/BIG-bench).
• Scores either as an average of more fine-grained (see Interaction cell).</td></tr>
</table>

Table 1: Road map to interactive language modeling.

pute the loss) for each sample by adding the log probabilities of each element in the sample. We explain the rationale behind this in Appendix A.

4.2 The Student

As the teacher is the main focus of our work, we choose to keep the student side simple. We represent the student as a causal Transformer LM, that we train on the data that it receives from the teacher.

4.3 The Interaction

Following Table 1, the teacher sends all selected data to the student at once. The student uses this data to train its LM and takes an exam after a predefined number of updates. The average exam score is sent back to the teacher as feedback. We use the student's last model checkpoint to compute the scores (as opposed to the best checkpoint on a validation set), to ensure that the learning signal for the teacher is restricted to the student's performance on the exam set, i.e., we do not expect teachers to reverse the learning process of the students (just like caregivers cannot do this for their children).

4.4 The Environment

Following Table 1, we design a number of artificial languages to test our approach on (see Section 5 for details). Using artificial languages is a well-tested approach to study the behavior of neural networks (e.g., Batali, 1994; Wiles and Elman, 1995; Rodriguez et al., 1999; Gers and Schmidhuber, 2001; Rodriguez, 2001; Hupkes et al., 2018; Lake and Baroni, 2018; Saxton et al., 2019; Hupkes et al., 2020; Rodríguez Luna et al., 2020; van der Wal et al., 2020; Chaabouni et al., 2021; Dagan et al., 2021). Using artificial languages gives us the control we need to design our experiments in such a way that we can correctly interpret the results.

4.5 The Exam

The exam is a held-out set over which we compute the student's perplexity. The details of the exam are task dependent and we discuss these next.

5 Experiments

We test our proposed setup on a number of settings and tasks, that we describe in this section.

5.1 Task 1 – Teaching Different Domains

For this task we design a language consisting of two strictly separated vocabularies, loosely representing two different domains in natural language. Specifically, $V_1 = \{a, b, c, d, e, f, g, h, i, j\}$, and $V_2 = \{k, l, m, n, o, p, q, r, s, t\}$. We construct sentences by randomly sampling from these sets. Sentences consist either of tokens only from V_1 or of tokens only from V_2. Sentences have an equal length of 10 tokens each. Half of the data set that the teacher can choose from consists of V_1 sentences, the other half consists of V_2 sentences. The teacher's LM is trained on a similarly constructed data set, yet consisting of different sentences. The student's exam set consists of sentences from only one of the vocabularies, V_1 in our case. These are different sentences than in the training set, i.e., the teacher cannot simply sample the exam set to train the student. Hence, the optimal teaching strategy is to present the student with sentences from the exam vocabulary. We confirm this in our baseline experiments that we present in Section 5.4.

5.2 Task 2 – Teaching Different Structures

For this task we do not use different vocabularies, but different sentence structures. All our sentences are constructed with V_1 and are between 2 and 10 tokens long. We use two different structures: single repetitions and double repetitions. In the case of the single repetitions two identical tokens never occur next to each other, whereas in the case of double repetitions tokens are sampled in pairs:
Structure 1 - Single repetitions: $(xy)^n$
Structure 2 - Double repetitions: (xx) or $(xxyy)^n$
The data set that the teacher can sample from consists for 20% of sentences with Structure 1 and for 80% of sentences of Structure 2. The exam set consists of sentences with Structure 1. We opt for this way of splitting the data, as we found that a student performs quite well when trained on data consisting half of Structure 1 and half of Structure 2. Having an unequal split thus allows us to make sure that we can appropriately distinguish a learned didactic approach from a random one. For this task the optimal teaching strategy is to select sentences with the exam structure, as we confirm with our baseline experiments that we present in Section 5.4.

5.3 Training Details

The teacher LM is trained on 100 unique sentences till convergence. The dataset the teacher can sample from for the student consists of 100 different unique sentences. The exam consists of 10 unique sentences and we set the teacher budget to 10 as well. We run our experiments with five different random seeds and report the averages and standard

deviations. We use the negative perplexity of the student on the exam as reward for the teacher. We experiment with two different sentence embeddings for the teacher: average word embeddings and the average of the last hidden layer. We train students for a predefined number of steps that we determine by inspecting the loss and perplexity curves of training an LM once before the actual experiments. We base the threshold on when a student LM starts to overfit, so that a teacher can get clear feedback signals. We set this value to 400 for Task 1 and 300 for Task 2. Automatically determining when the students stops training is an important avenue for future work (Table 1). We use Fairseq's (Ott et al., 2019) `transformer_lm`[2] for the implementation of the Transformer LMs. We use up to four GPUs with 32 GB RAM per experiment. The exact number depends on the number of students per teacher, as we can fit up to 6 students on a single GPU due to our multiprocessing implementation.

5.4 Baseline experiments

We run three baseline experiments with three different didactic strategies: an *oracle*, *random*, and *worst case* strategy. We run the baselines for five different random seeds. In each experiment, we randomly select data according to the teacher budget. We do this five times and each time train a student LM with the selected data. The difference between baselines is the type of data that can be selected. For the oracle baseline we only select sentences that consist of the exam vocabulary (Task 1) or structure (Task 2). For the random baseline we randomly select sentences. For the worst case baseline all sentences that we select are from a different vocabulary or structure than the exam sentences.

6 Results

6.1 Task 1 – Different Domains

6.1.1 Baseline Results

In Table 2 we present the results for the baseline experiments for Task 1. We report the averages and standard deviations of the perplexity on the exam set and the fraction of training sentences that consisted of the exam vocabulary. For space reasons, we report the results for two seeds per baseline: the seed with the best average perplexity and the worst. The results for all fives seeds are given in Appendix B. There we also present scores for the n-gram overlap between the selected training set and the exam set. The results are as expected. The oracle baseline gives the best results, followed by the random and worst case baseline respectively.

Type	Seed	Avg Perplexity	Avg train from test
Rand.	B	160.9 ± 217.7	0.54 ± 0.16
	W	742.5 ± 159.8	0.50 ± 0.17
Orac.	B	14.99 ± 5.364	1.00 ± 0.00
	W	68.95 ± 87.49	1.00 ± 0.00
Worst case	B	$4.78e4 \pm 2.67e4$	0.00 ± 0.00
	W	$8.46e4 \pm 4.69e4$	0.00 ± 0.00

Table 2: Baseline results Task 1. Averages and standard deviations reported based on five runs per seed. *Rand* is Random, *Orac* is Oracle, *B* is Best and *W* is Worst.

6.2 Results of Training the Teacher

In Figure 3 we present the results for Task 1 for different numbers of students per teacher.[3] The teacher's didactic strategy correctly converges to the oracle baseline. There is a clear difference between different sentence embeddings (Section 4.1.1). Both embedding types are converging, but the average hidden layer embeddings are clearly superior. We investigate this further by plotting the t-SNE embeddings (Van der Maaten and Hinton, 2008) of the different sentence embeddings in Figure 4. To prepare for Task 2, we also plot the embeddings of Task 2. The hidden layer sentence embeddings result in the clearest separation between sentences from different vocabularies or structures. Especially for Task 2, where we use the same vocabulary, this is unsurprising. From now on we opt for these sentence embeddings. Based on the results for Task 1 we opt for 12 students per teacher as a good trade-off between computational cost and convergence stability for Task 2.

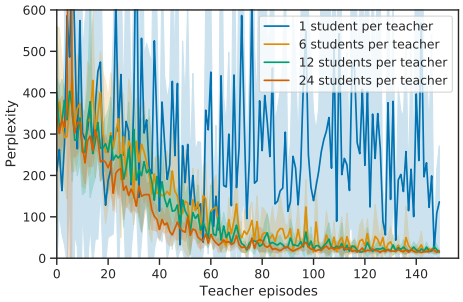
6.3 Task 2 – Different Structures

6.3.1 Baseline Results

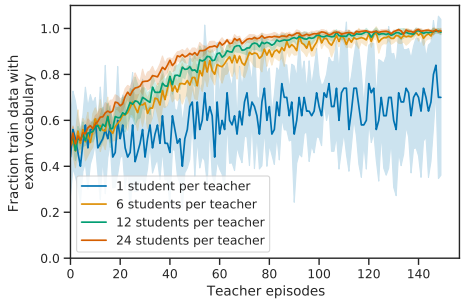
We present the results for the baseline results for Task 2 in Table 3. Again we report the results for the best and the worst seed. Full results are available in Appendix C. Similarly to the results for Task 1, we confirm that the oracle baseline performs strongest, followed by the random and worst case baseline respectively.

[2] https://fairseq.readthedocs.io/en/latest/command_line_tools.html

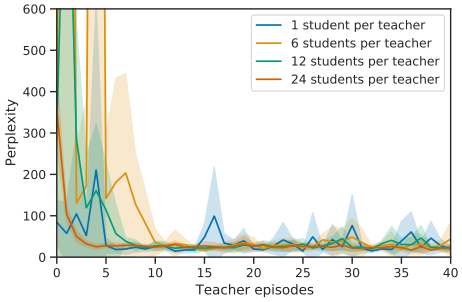
[3] We present plots for the n-gram overlap in Appendix D.



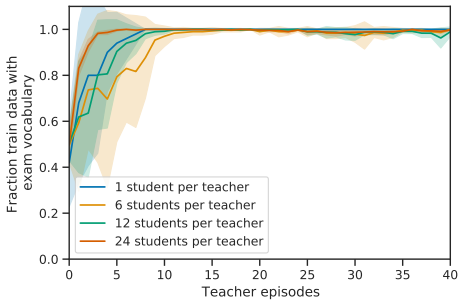
(a) Perplexity of the student on the exam data over different episodes. Average word embedding as input to the teacher's policy.



(b) Fraction training data with the exam vocabulary over different episodes. Average word embedding as input to the teacher's policy.

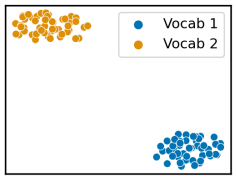


(c) Perplexity of the student on the exam data over different episodes. Average last hidden layer as input to the teacher's policy.

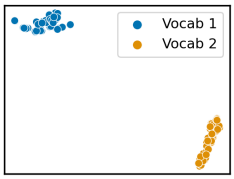


(d) Fraction training data with the exam vocabulary over different episodes. Average last hidden layer as input to the teacher's policy.

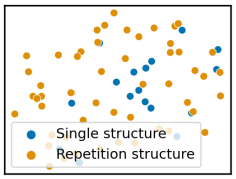
Figure 3: Results Task 1 – Different domains. Plots for different numbers of students per teacher. Results per setting reported as average and standard deviation over five random seeds. x-axis of lower plots bound to 40 as the teacher had already converged by then.



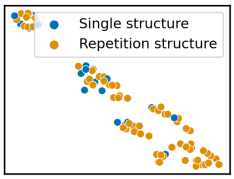
(a) Task 1 - Different vocabularies. Sentence embedding is average word embeddings.



(b) Task1 - Different vocabularies. Sentence embedding is average last hidden layer.

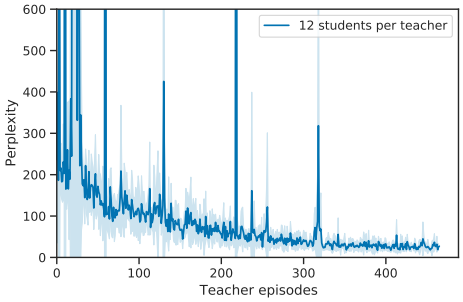


(c) Task 2 - Different structures. Sentence embedding is average word embeddings.

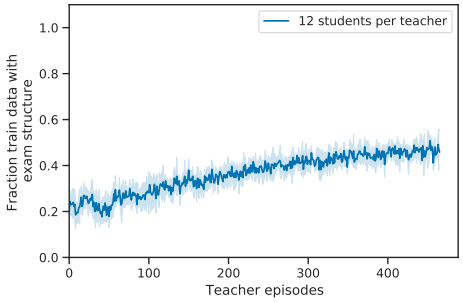


(d) Task 2 - Different structures. Sentence embedding is average last hidden layer.

Figure 4: T-SNE plots for different sentence representations for different tasks.



(a) Perplexity of the student on the exam data over different episodes.



(b) Fraction training data with the exam structure over different episodes.

Figure 5: Results Task 2 – Plots for 12 students per teacher. Results per setting reported as average and standard deviation over five random seeds.

Type	Seed	Avg Perplexity	Avg train from test
Rand.	B	119.0 ± 56.48	0.18 ± 0.04
	W	342.1 ± 241.4	0.12 ± 0.08
Orac.	B	6.821 ± 0.619	1.00 ± 0.00
	W	9.431 ± 3.057	1.00 ± 0.00
Worst Case	B	299.6 ± 124.2	0.00 ± 0.00
	W	595.3 ± 297.9	0.00 ± 0.00

Table 3: Baseline results Task 2. Averages and standard deviations reported based on five runs per seed. *Rand* is Random, *Orac* is Oracle, *B* is Best and *W* is Worst.

6.3.2 Results of Training the Teacher

In Figure 5 we present the results for Task 2.[4] Again we see that the teacher learns to gradually converge to the oracle teaching strategy, although convergence is less fast than for Task 1; we do not achieve full convergence in the number of training episodes that we run these experiments for. We postulate that this can be explained by the differences we found in Figure 4. The differences in sentence embeddings between the two different structures are clearly less apparent than between the sentences from two vocabularies. This indicates the importance of good sentence embeddings in future work. Moreover, as stated in Section 6.3, we found that transmitting roughly 50% of Structure 1 and 50% of Structure 2 also already leads to good performance. Therefore, the teacher likely needs to learn from a less distinct learning signal than in Task 1.

7 Implications and Outlook

We successfully took the first steps on our proposed road map. Here we want to share our learnings and the limitations of the current setup to help future research to take the next steps on the road map.

The importance of designing experiments with interpretable outcomes. We designed our experiments such that we knew the teacher's oracle strategy, which allowed us to properly test our setup. However, in designing our experiments we found that finding such settings is non-trivial. For example, in a task that contains a language with multiple structures, a student might unexpectedly learn information from structure 1 that also proves useful for structure 2. This might be acceptable if one's only objective is to obtain a good performance. However, in our case it is critical to be able to know that a teacher is "right for the right reasons", which motivated our choices for the tasks and languages.

The teacher's budget. Following our objective, we designed our experiments in such a way that the teacher was given a budget that limits the amount of data it can send to the student. As mentioned in Section 5.3, we confirmed that the student's learning converges with this budget. In follow up work we plan to investigate the importance of different budgets in more detail. One interesting direction is to give the teacher a flexible budget, i.e., such that a teacher could decide to stop training if it deems it no longer necessary for the student.

Computational complexity. Apart from the multiprocessing setup that allows us to train multiple students on a single GPU, we did not yet focus on the computational complexity of our approach. In the current setup many student language models need to be trained for a single teacher. In our case we deem this justifiable as we are just at the start of the road map. Moreover, once a teacher model is trained, it can be used for many different purposes. However, in future work we hope to focus on decreasing the computational complexity of our approach. One promising avenue to do this is by optimizing the learning process of the student.

8 Conclusion

In this paper we pioneered the space of interactive language modeling, motivated by the observation that current state-of-the-art LMs are trained in a very unnatural way, from the perspective of human language acquisition. Moreover, an interactive approach has the potential to make LMs more efficient and adaptable. Specifically, we proposed a teacher-student loop, in which the teacher is inspired by the caregiver and the student resembles the child in the human language acquisition. We presented a road map that details the steps towards interactive language modeling for each of the components of the teacher-student loop. We led by example and took the first steps on this road map, leading to a tangible proof of concept of our proposal. As such, we structured the space of interactive language modeling and aim to inspire a larger research agenda on interactive language modeling.

9 Ethical Impact Statement

At this point we use artificial language data only, for which we do not see any direct negative impli-

[4] We present plots for the n-gram overlap in Appendix E.

cations. As we move towards using real data sets, it is necessary to be aware of potential biases with these data sets. One needs to ensure that the data is not biased towards any (protected) group to avoid any harm. Currently, much of the NLP research focuses on English as its language of interest. Our approach is not bound to any language in particular and can even be used to improve language learning in a low resource setting. Once the models achieve human like performance and are used for downstream tasks and applications it is necessary to explicitly state that language is produced by an artificial language model. However, as with all language models, misuse can still happen and it is our responsibility as a research community, amongst others, to spend effort on making users aware of these possibilities.

References

John Batali. 1994. Artificial evolution of syntactic aptitude. In *Proceedings from the Sixteenth Annual Conference of the Cognitive Science Society*, pages 27–32. Lawrence Erlbaum Associates Hillsdale, NJ.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Jerome Bruner. 1985. Child's talk: Learning to use language. *Child Language Teaching and Therapy*, 1(1):111–114.

Rahma Chaabouni, Roberto Dessì, and Eugene Kharitonov. 2021. Can transformers jump around right in natural language? assessing performance transfer from scan. *arXiv preprint arXiv:2107.01366*.

Eve V Clark. 2018. Conversation and language acquisition: A pragmatic approach. *Language Learning and Development*, 14(3):170–185.

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. 1996. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145.

Alejandrina Cristia, Emmanuel Dupoux, Nan Bernstein Ratner, and Melanie Soderstrom. 2019. Segmentability differences between child-directed and adult-directed speech: A systematic test with an ecologically valid corpus. *Open Mind*, 3:13–22.

Gautier Dagan, Dieuwke Hupkes, and Elia Bruni. 2021. Co-evolution of language and agents in referential games. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2993–3004, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. Banditsum: Extractive summarization as a contextual bandit. *arXiv preprint arXiv:1809.09672*.

Liat Ein Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active learning for bert: An empirical study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Felix A Gers and Jürgen Schmidhuber. 2001. LSTM recurrent networks learn simple context-free and context-sensitive languages. *Neural Networks, IEEE Transactions on*, 12(6):1333–1340.

Steven Gillis and Annemarie Schaerlaekens. 2000. *Kindertaalverwerving: Een handboek voor het Nederlands*.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: how do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. 2021. Lifelong pretraining: Continually adapting language models to emerging corpora. *arXiv preprint arXiv:2110.08534*.

Julia Kiseleva, Ziming Li, Mohammad Aliannejadi, Shrestha Mohanty, Maartje ter Hoeve, Mikhail Burtsev, Alexey Skrynnik, Artem Zholus, Aleksandr Panov, Kavya Srinet, et al. 2021. Neurips 2021 competition iglu: Interactive grounded language understanding in a collaborative environment. *arXiv preprint arXiv:2110.06536*.

Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without

replacement. In *International Conference on Machine Learning*, pages 3499–3508. PMLR.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 4487–4499.

Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Sebastian Ruder, Dani Yogatama, et al. 2021. Pitfalls of static language modelling. *arXiv preprint arXiv:2102.01951*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. 2019. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*.

Mitja Nikolaus and Abdellah Fourtassi. 2021. Modeling the interaction between perception-based and production-based learning in children's early acquisition of semantic knowledge.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

J. Rapin and O. Teytaud. 2018. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad.

Roi Reichart, Katrin Tomanek, Udo Hahn, and Ari Rappoport. 2008. Multi-task active learning for linguistic annotations. In *Proceedings of ACL-08: HLT*, pages 861–869.

Paul Rodriguez. 2001. Simple recurrent networks learn context-free and context-sensitive languages by counting. *Neural computation*, 13(9):2093–118.

Paul Rodriguez, Janet Wiles, and Jeffrey L Elman. 1999. A recurrent neural network that learns to count. *Connection Science*, 11(1):5–40.

Diana Rodríguez Luna, Edoardo Maria Ponti, Dieuwke Hupkes, and Elia Bruni. 2020. Internal and external pressures on language emergence: least effort, object constancy and frequency. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4428–4437, Online. Association for Computational Linguistics.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*.

Laura A Shneidman and Susan Goldin-Meadow. 2012. Language input and acquisition in a mayan village: How important is directed speech? *Developmental science*, 15(5):659–673.

Katharina Stein, Leonie Harter, and Luisa Geiger. 2021. Shapelurn: An interactive language learning game with logical inference. In *Proceedings of the First Workshop on Interactive Learning for Natural Language Processing*, pages 16–24.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Oskar van der Wal, Silvan de Boer, Elia Bruni, and Dieuwke Hupkes. 2020. The grammar of emergent languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3339–3359, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Tim Vieira. 2014. Gumbel-max trick and weighted reservoir sampling.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R Bowman. 2020. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Janet Wiles and Jeff Elman. 1995. Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent networks. In *Proceedings of the seventeenth annual conference of the cognitive science society*, s 482, page 487. Erlbaum Hillsdale, NJ.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020. Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6095–6104.

Wei Zhang, Wei Wei, Wen Wang, Lingling Jin, and Zheng Cao. 2021. Reducing bert computation by padding removal and curriculum learning. In *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 90–92. IEEE.

A Computing the Probability of a Top-K Sample

Our objective is to find the (log) probability of sampling the subset $(i_1, ...i_K)$ from $\{1, ..., N\}$ *without* replacement from the categorical probability $(p_1, ..., p_N)$.

Let us first consider sampling K elements from the $\{1, ..., N\}$ *with* replacement. In that case

$$p(i_1, ..., i_K) = \prod_{k=1}^{K} p_{i_k}. \tag{1}$$

If we allow for all possible permutations of observing $(i_1, ..., i_K)$ we get

$$p(i_1, ..., i_K) = C \prod_{k=1}^{K} p_{i_k}, \tag{2}$$

where $C = K!$.

To go from sampling *with* replacement, to sampling *without* replacement, we consider event A = "all sampled elements $(i_1, ..., i_K)$ are unique". Then

$$p_{\text{w/o replacement}}(i_1, ..., i_K) = p_{\text{w/ replacement}}(i_1, ..., i_K|A). \tag{3}$$

Applying Bayes Rule gives us:

$$p_{\text{w/o replacement}}(i_1, ..., i_K) = \frac{p_{\text{w/ replacement}}(A|i_1, ..., i_K) p_{\text{w/ replacement}}(i_1, ..., i_K)}{p_{\text{w/ replacement}}(A)}. \tag{4}$$

As in our case all samples in $(i_1, ..., i_K)$ are unique we know that

$$p_{\text{w/ replacement}}(A|i_1, ..., i_K) = 1. \tag{5}$$

Combining this with Equation 2 gives us

$$p_{\text{w/o replacement}}(i_1, ..., i_K) = \frac{C \prod_{k=1}^{K} p_{i_k}}{p(A)}, \tag{6}$$

and thus

$$p_{\text{w/o replacement}}(i_1, ..., i_K) \propto \prod_{k=1}^{K} p_{i_k}, \tag{7}$$

and

$$\log p_{\text{w/o replacement}}(i_1, ..., i_K) \propto \sum_{k=1}^{K} \log p_{i_k}. \tag{8}$$

From an implementation perspective this this boils down to the following steps:

1. We compute the scores per sentence.
2. We sample K sentences without replacement, using the GumbelTopK trick.
3. We compute the log probabilities for each score: $\log \text{softmax}(scores)$.
4. We compute the log probability of our sample by adding the log probabilities of the elements in our sample, according to Equation 8.

A.1 Comparison to Prior Work

Our problem of sampling K sentences as a single action is similar to the problem formulation of using Reinforcement Learning for extractive summarization to optimize for Rouge (Lin, 2004) directly. In this setting K sentences need to be selected from a document. This results in a very large search space. Narayan et al. (2018) limit the search space by first selecting n sentences that have a high Rouge score. Then all possible summaries are made with these n sentences. These summaries are ranked according to their Rouge scores and the top K sentences are taken as action. This approach

has the disadvantage that it limits the search space heuristically, which does not guarantee that the best summary is found. Dong et al. (2018) frame the problem as a contextual bandit problem, which allows them to sample from the true action space. We choose our approach as it is intuitive, simple and effective.

B Additional Results Baseline Experiments Task 1

In Table 4 we present the results for our baseline runs on all five seeds.

Baseline	Seed	Avg Perplexity	Avg train from test	Avg unigram overlap	Avg bigram overlap	Avg trigram overlap
Random	6639	193.9 ± 100.3	0.46 ± 0.14	0.46 ± 0.14	0.278 ± 0.07	0.023 ± 0.009
	7519	683.1 ± 634.3	0.52 ± 0.15	0.52 ± 0.15	0.291 ± 0.10	0.030 ± 0.010
	1007	742.5 ± 159.8	0.50 ± 0.17	0.50 ± 0.17	0.298 ± 0.10	0.035 ± 0.014
	4520	160.9 ± 217.7	0.54 ± 0.16	0.54 ± 0.16	0.327 ± 0.09	0.035 ± 0.025
	4527	307.1 ± 295.1	0.58 ± 0.17	0.58 ± 0.17	0.349 ± 0.10	0.035 ± 0.014
Oracle	6639	14.99 ± 5.364	1.00 ± 0.00	1.00 ± 0.00	0.551 ± 0.06	0.072 ± 0.029
	7519	44.37 ± 58.94	1.00 ± 0.00	1.00 ± 0.00	0.611 ± 0.02	0.085 ± 0.017
	1007	68.95 ± 87.49	1.00 ± 0.00	1.00 ± 0.00	0.598 ± 0.02	0.077 ± 0.025
	4520	15.65 ± 4.616	1.00 ± 0.00	1.00 ± 0.00	0.578 ± 0.02	0.087 ± 0.028
	4527	23.66 ± 21.44	1.00 ± 0.00	1.00 ± 0.00	0.624 ± 0.02	0.095 ± 0.019
Worst case	6639	$8.46e4 \pm 4.69e4$	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	7519	$7.03e4 \pm 3.73e4$	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	1007	$8.17e4 \pm 4.26e4$	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	4520	$4.78e4 \pm 2.67e4$	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	4527	$6.69e4 \pm 1.98e4$	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00

Table 4: Baseline results for Task 1. Different domains. Averages and standard deviations reported based on five runs per seed.

C Additional Results Baseline Experiments Task 2

In Table 5 we present the results for our baseline runs on all five seeds.

D Additional Results Task 1

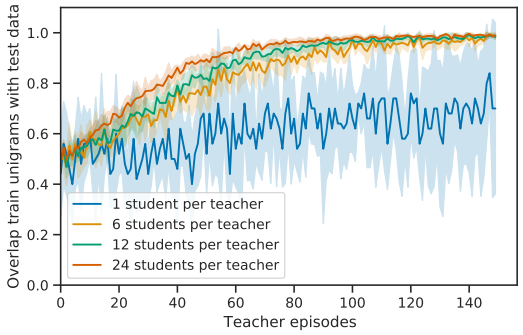
In this section we present the plots for the n-gram overlap for Task 1 in Figures 6 and 7.

E Additional Results Task 2

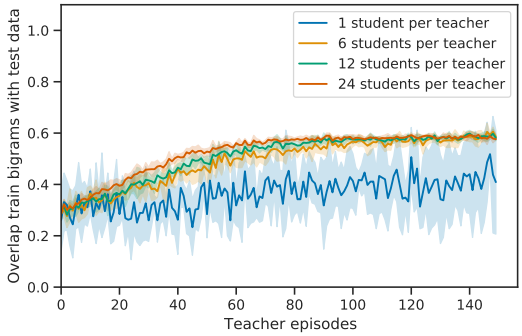
In this section we present the plots for the n-gram overlap for Task 2 in Figure 8.

Baseline	Seed	Avg Perplexity	Avg train from test	Avg unigram overlap	Avg bigram overlap	Avg trigram overlap
Random	6639	119.0 ± 56.48	0.18 ± 0.04	1.00 ± 0.00	0.401 ± 0.033	0.030 ± 0.020
	7519	162.8 ± 201.9	0.24 ± 0.05	1.00 ± 0.00	0.408 ± 0.044	0.035 ± 0.038
	1007	234.1 ± 192.0	0.24 ± 0.12	1.00 ± 0.00	0.414 ± 0.034	0.034 ± 0.020
	4520	161.7 ± 190.6	0.22 ± 0.04	1.00 ± 0.00	0.410 ± 0.023	0.038 ± 0.033
	4527	342.1 ± 241.4	0.12 ± 0.08	1.00 ± 0.00	0.348 ± 0.024	0.013 ± 0.017
Oracle	6639	6.973 ± 1.534	1.00 ± 0.00	1.00 ± 0.00	0.720 ± 0.044	0.151 ± 0.022
	7519	7.626 ± 2.298	1.00 ± 0.00	1.00 ± 0.00	0.682 ± 0.056	0.177 ± 0.033
	1007	7.895 ± 1.106	1.00 ± 0.00	1.00 ± 0.00	0.726 ± 0.045	0.207 ± 0.025
	4520	6.821 ± 0.619	1.00 ± 0.00	1.00 ± 0.00	0.740 ± 0.073	0.197 ± 0.054
	4527	9.431 ± 3.057	1.00 ± 0.00	1.00 ± 0.00	0.700 ± 0.056	0.174 ± 0.017
Worst case	6639	595.3 ± 297.9	0.00 ± 0.00	1.00 ± 0.00	0.326 ± 0.026	0.00 ± 0.00
	7519	317.2 ± 235.8	0.00 ± 0.00	1.00 ± 0.00	0.311 ± 0.018	0.00 ± 0.00
	1007	508.1 ± 155.7	0.00 ± 0.00	1.00 ± 0.00	0.345 ± 0.017	0.00 ± 0.00
	4520	299.6 ± 124.2	0.00 ± 0.00	1.00 ± 0.00	0.310 ± 0.027	0.00 ± 0.00
	4527	432.8 ± 72.05	0.00 ± 0.00	1.00 ± 0.00	0.330 ± 0.035	0.00 ± 0.00

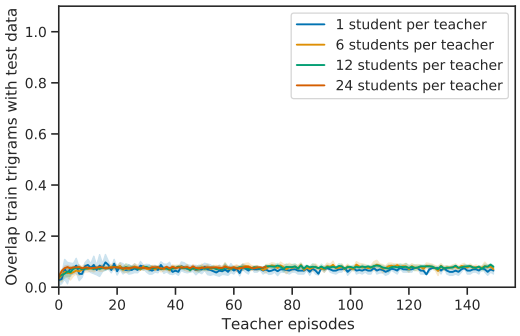
Table 5: Baseline results for Task 2. Different structures. Averages and standard deviations reported based on five runs per seed.



(a) Unigram overlap between train and test data.

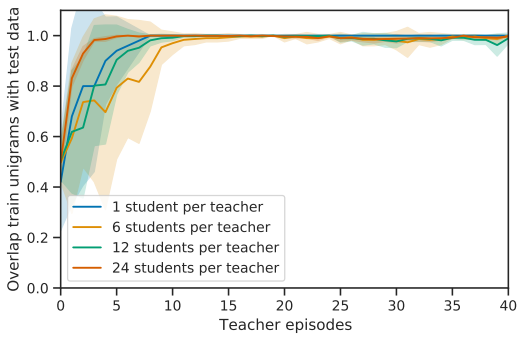


(b) Bigram overlap between train and test data.

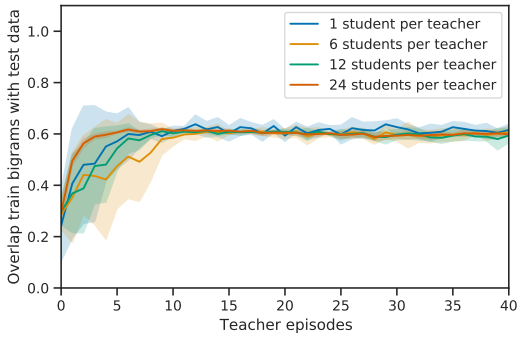


(c) Trigram overlap between train and test data.

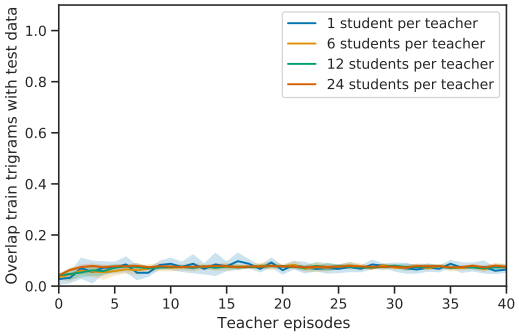
Figure 6: Additional results Task 1 – Different domains. Plots for different numbers of students per teacher. Results per setting reported as average and standard deviation over five random seeds. Average word embedding as sentence embeddings.



(a) Unigram overlap between train and test data.

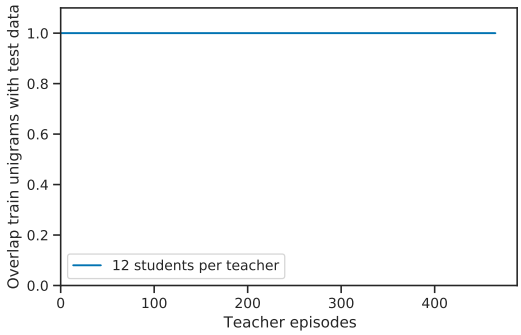


(b) Bigram overlap between train and test data.

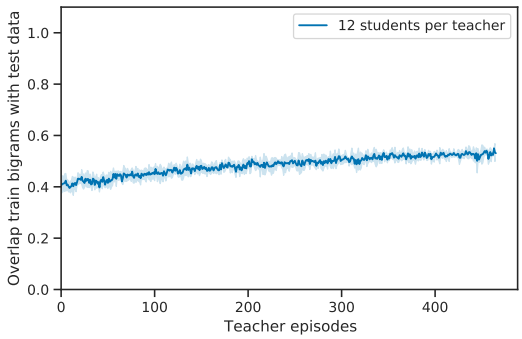


(c) Trigram overlap between train and test data.

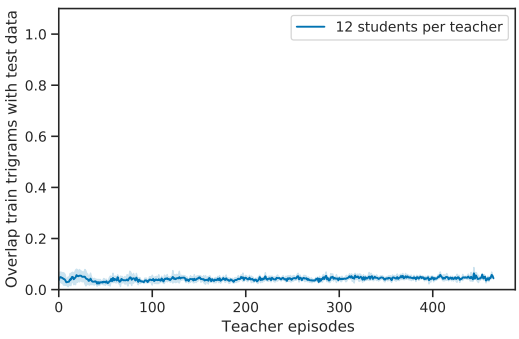
Figure 7: Additional results Task 1 – Different domains. Plots for different numbers of students per teacher. Results per setting reported as average and standard deviation over five random seeds. Average hidden layer embedding as sentence embeddings.



(a) Unigram overlap between train and test data.



(b) Bigram overlap between train and test data.



(c) Trigram overlap between train and test data.

Figure 8: Additional results Task 2 – Different structures. Results per setting reported as average and standard deviation over five random seeds.