Modeling Spaced Repetition with LSTMs

Jakub Pokrywka[1][a], Marcin Biedalak[2], Filip Graliński[1][b] and Krzysztof Biedalak[2]
[1] *Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland*
[2] *SuperMemo World*
{jakub.pokrywka, filip.gralinski}@amu.edu.pl,{marcin.biedalak, krzysztof.biedalak}@supermemo.com

Keywords: spaced repetition, LSTM, metalearning

Abstract: Spaced repetition is a human learning technique focused on optimizing time intervals between a student's repetitions of the same information items. It is designed for the most effective long-term high-retention knowledge acquisition in terms of a student's time spent on learning. Repetition of an information item is performed when its estimated recall probability falls to the required level. Spaced repetition works particularly well for itemized knowledge in areas requiring high-volume learning like languages, computer science, medicine, etc. In this work, we present a novel machine-learning approach for the prediction of recall probability developed using the massive repetition data collected in the SuperMemo.com learning ecosystem. The method predicts the probability of remembering an item by a student using an LSTM neural network. In our experiments, we observed that applying the spaced repetition research expert algorithms (Woźniak et al., 2005), like imposing the negative exponential function as the output forgetting curve, increases the LSTM model performance. We analyze how this model compares to other machine-learning or expert methods such as the Leitner method, XGBoost, half-life regression, and the spaced repetition expert algorithms. We found out that the choice of evaluation metric is crucial. Furthermore, we elaborate on this topic, finally selecting macro-average MAE and macro-average Likelihood for the primary and secondary evaluation metrics.

1 INTRODUCTION

Spaced repetition is the idea to improve learning process for humans by optimizing the time intervals between which the same material, for instance a word in a foreign language (in general: a repetition item), is presented to the user. E-learning systems based on spaced repetition are used for courses based on a large number of atomic items, for instance in learning foreign (or programming) languages (especially their vocabulary) or acquiring fact-based knowledge.

The idea of spaced-repetition software was pioneered by Piotr Woźniak and SuperMemo with a number of expert algorithms. In this paper, we train a number of machine-learning-based systems, using massive repetition data obtained through the courtesy of SuperMemo.com platform, and compare them against simple baselines and the original spaced-repetition expert algorithms (Woźniak, 1990; Woźniak et al., 1995).

The contributions of this paper are as follows:

1. we propose a methodology for creating future-proof challenges from real-world data for training and testing spaced-repetition systems;
2. we discuss evaluation methodology and give motivation for using a new evaluation metric;
3. we propose a novel approach of a LSTM neural network with exponential decay and compare it to several baselines.

2 SUPERMEMO RESEARCH

SuperMemo is the world pioneer in applying optimized spaced repetition to computer-aided learning (Woźniak, 2018a). The name SuperMemo encompasses the method, software and company. In 1982, Piotr Woźniak, then a student of molecular biology, started experiments which led to the formulation of his first spaced repetition algorithm in 1985. In 1987, he created the first SuperMemo computer program. It applied the so-called SM-2 algorithm(Woźniak, 1990) which was later made public and has been used by other apps, including Anki, ever since. In the following years, Woźniak kept improving his expert

[a] https://orcid.org/0000-0003-3437-6076
[b] https://orcid.org/0000-0001-8066-4533

algorithm. Successive versions adapted to the actual memory retention measured individually for each user, thus allowing for truly individualized learning. Independently of this, Woźniak developed his theory of two components of memory (Woźniak et al., 1995), which was fully applied in the SM-17 algorithm in 2016.

While optimizing the machine learning algorithms described in this paper, we successfully used key elements of Woźniak's research. In order to smooth the recall probability predictions yielded for increasing intervals, we forced the LSTM networks (see Section 8.7) to apply the negatively exponential function which, as proposed by Woźniak, represents the shape of the forgetting curve (Woźniak et al., 2005):

$$R = e^{\ kt=S};$$

where:

- t — time,
- R — probability of recall at time t,
- S — stability expressed by the inter-repetition interval that results in retrievability of 90% (i.e. $R = 0{:}9$),
- k — constant independent of stability.

To some surprise, it not only matched the original LSTM results but also slightly improved the algorithm metrics.

3 SUPERMEMO.COM ECOSYSTEM AND DATA

For training and testing the machine learning algorithms described in this paper, we obtained repetition data collected by the SuperMemo.com online and apps learning ecosystem. SuperMemo.com features over 250 ready language courses for 23 different languages in the premium version and allows users to learn from user-generated courses for free. SuperMemo.com is often applied by users to learn sciences requiring high-volume learning, including computer science, programming and medicine.

SuperMemo.com courses differentiate between presentation and repetition content. Presentation pages are used for explaining the material learned. When users progress through a course, presentation items are shown once by default. They may include comments, complex texts or even parts of a full feature interactive movie. Repetition items (exercises) contain atomic questions or tests which are then scheduled in repetitions according to the SuperMemo algorithm. While learning languages, these are typically used to memorize vocabulary and grammar. When learning programming, exercises can be used to master coding rules and patterns (see Figure 1). In general, for optimum review scheduling and learning, repetition items are recommended to meet the *minimum information principle* (Woźniak, 1999) (i.e. should be atomic and as simple as possible).

SuperMemo repetition items typically test active production. Passive knowledge, like developed in multiple choice tests, is considered to be a different, limited competence. Therefore, unlike in other popular e-learning applications which often shuffle the same content along different types of multiple choice tests during a session, SuperMemo exercises are mostly question and answer pairs which are stable in their form. Each exercise, irrespective of whether it is active or passive (see Figure 2), is treated as a separate item with its own learning characteristics and history. Each repetition is rated once on the first contact during a session.

Exercises are rated on a 3-grade scale: *I know*, *Almost*, *Don't know*. The first two are both positive grades meaning that the information is still remembered, with the difference that answers rated *I know* are not asked again in the same session, while those rated *Almost* can be drilled until they are recalled successfully. Based on the history of repetitions and grades, the SuperMemo algorithm proposes the next repetition for the day when the probability of recall by the user is expected to fall to 90% (see Figure 3).

The SuperMemo algorithm develops and maintains separate memory models for every user and course. Each exercise is scheduled for repetitions so as to statistically reach the expected level of retention.

4 RELATED WORK

Half-life regression is a model of space repetition, a modification of linear/logistic regression taking into account the forgetting curve (Settles and Meeder, 2016). Note that Duolingo, the system for which half-life regression was initially proposed, is based on an approach different from SuperMemo — a gold-standard value does not have to be 0 or 1, it is usually a fraction representing the percentage of successful attempts during a single session. In SuperMemo, a simpler model is assumed (following the minimum information principle), a model that can handle a larger variety of courses.

Deep reinforcement learning have been also applied to the problem of planning spaced repetition (Upadhyay et al., 2018; Yang et al., 2020; Sinha, 2019). The main drawback of reinforcement learning

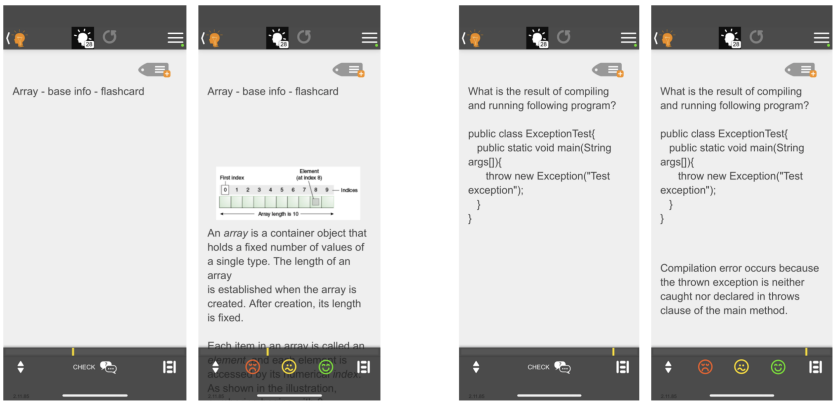

Figure 1: Pages from general computer science and Java 8 programming courses, source: SuperMemo application.

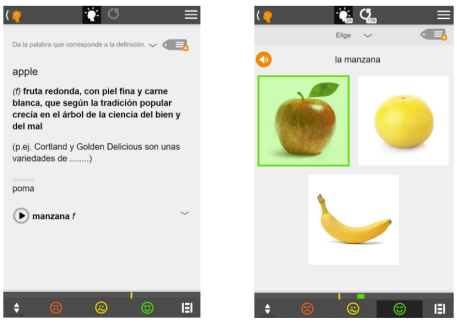

Figure 2: Active and passive exercises in SuperMemo are separate items for repetition scheduling, source: SuperMemo application.

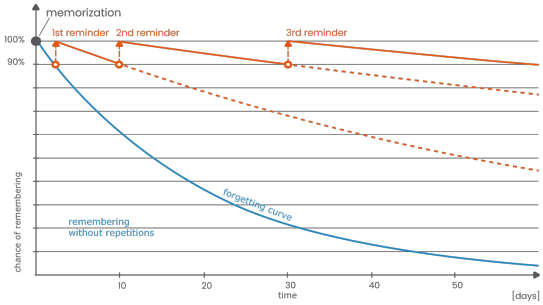

Figure 3: Forgetting curves applied in learning, source: www.supermemo.com

in this context is that it requires simulated environments for training and evaluation. In this paper, we opt for a more practical option of limiting ourselves to the paradigm of supervised learning.

5 DATA SET

The data set is based on real retention data from the SuperMemo application.

5.1 Assumptions

In order to make the challenge harder and to simulate cases when a new user joins the system or a new course is created, the train/dev/test split was prepared in such a way that no user and no course is shared between the data subsets. To be more precise, the MD5 sum is calculated for each user and for each course, and users and courses are (separately) assigned to the training, development and test sets based on its checksums. This assignment has the following consequences:

- the train/dev/test split is pseudo-random, but *stable*, when the splitting script is re-run on a (possible larger data set), no course/user will change assignments,
- some data is "lost", e.g. a user-course pair for which the user is assigned to the train set and the course to the dev set will not be used,
- ...but on the other hand, we are avoiding unwanted data leakage between users and courses.

In other words, we pose the challenge as *meta-learning* problem. We do not want to learn features for specific users or courses, but rather learn *to learn* to predict retention probabilities even for *new* users (and courses).

For the development and testing sets, a single repetition is selected, again using a stable pseudo-random procedure based on MD5 checksums. All the repetitions up to this one are available, including repetitions related to other words (learning units), but the history *after* the target repetition is removed.

For the training set, simply all the repetitions are given.

The data set was prepared as a challenge on an internal instance of the Gonito platform for tracking evaluation results for machine-learning systems (Graliński et al., 2016).

Table 1: Basic statistics as regards the data set.

dataset	size	recall ratio
train	5757868	95.4%
dev	1152	89.1%
test	1611	88.6%

5.2 Statistics

Data used in this work was collected from users of SuperMemo.com platform where MongoDB is used to store information about every interaction with an item, see Table 1 for basic statistics. We obtained repetition date, previous and next interval set by algorithm, real interval from last repetition, grade. The task is to predict probability for the last grade.

6 EVALUATION METRICS

Selecting the most appropriate evaluation metric for spaced repetitions models is a challenging task. In (Settles and Meeder, 2016), three metrics were considered: mean absolute error (MAE), area under the ROC curve (AUC) and Spearman's half-life rank correlation. In the case of the SuperMemo learning system, the quality of probabilities is crucial, not just the accuracy of predicted classes (forgotten vs retained), as repetition intervals are directly based on probability thresholds (which can be customized by the user). Hence, we discarded Spearman's rank and AUC.

Apart from MAE, we measured the quality of probabilities using the *likelihood* metric, which is the geometric mean of probabilities assigned to the correct class by the model. This is a variant of log loss (if log loss is L, then likelihood is $1=e^{L}$), just made slightly easier to interpret for humans.

Contrary to MAE, likelihood (and log loss) are obviously highly sensitive to overconfident results. It takes one example in which 0 was returned as the probability for the positive class, or 1 for the negative one, for the likelihood metric to collapse to 0. Therefore, it is rational to assume some e and return at least e for an example with the presumably negative class and 1 e for an example with the positive one.

Another problem is that there is a significant imbalance in the training and testing sets — most samples (around 90 %) belong to the positive class (a user retained a given unit in the memory) and a simple null-model baseline (return 1.0 for MAE and 1 e for Likelihood) can lead to nominally high and hard-to-beat evaluation results. The approach we chose was to use the macro-average version of the metrics, i.e. the evaluation metric is calculated separately for the negative and the positive class and then averaged. This way, we attach the same significance to both classes, no matter their numbers.

Finally, we selected macro-avg MAE as the main evaluation metric and macro-avg likelihood as the secondary metric (as implemented as `MacroAvg/MAE` and `MacroAvg/Likelihood` in the GEval evaluation tool (Graliński et al., 2019)). Our decision was confirmed by the fact that all reasonable methods we tested beat the simple null-model baseline if macro-avg MAE was used.

Note that for MAE the lower results, the better, for Likelihood — the other way round.

For a different approach for the evaluation of spaced-repetition systems, based on the of idea of *algorithm contest*, see (Woźniak, 2018b).

7 EXISTING NON-ML METHODS

7.1 Null-Model Baseline

This is a simple baseline. During the inference, we just always return probability 0.89, i.e. the mean from training set for all samples.

7.2 Leitner

Leitner System is one of the simplest spaced repetitions methods, mainly used in flashcards (Leitner, 1999). The main idea is to repeat item on the next day after learning it, if the user recalls information correctly the interval is doubled. If not, the interval should be divided by 2 or set to 1.

7.3 SuperMemo Open-Source Algorithm SM-2

The first computer-based SuperMemo algorithm (Woźniak, 1990) which, for every item, tracks the number of times it has been successfully recalled and the interval (i.e. the number of days since the item has been repeated). For each review (attempt by the user to recall the item), the algorithm recalculates easiness factor (EF) based on the self-evaluated grade and sets the date for the next repetition. In this work, we recalculate probability by taking interval from algorithm and comparing it against the forgetting curve.

7.4 SuperMemo Expert Algorithm SM-17

The SM-17 algorithm, developed in the years 2014-16, applies the two component model of memory (Woźniak et al., 1995). Starting from a common memory model, SM-17 then stores and updates the DSR (difficulty, stability, retrievability) matrices with parameters individual for each user. Hill-climbing algorithms are used to find the best estimation of an expected forgetting curve.[1]

8 MACHINE LEARNING METHODS

We used the following machine learning methods: logistic regression, feed-forward neural network, half-life regression, gradient boosting trees, and recurrent neural network (RNN). Some of them incorporate exponential decay from the forgetting curve assumption. This may help in training, but more importantly, it does not lead to a counter-intuitive result when the probability of recalling an item increases with time when a student does not study it. For all methods, we take the maximum item history sequence of 40 most recent repetitions. This is necessary for all methods, but RNN. Due to easier batching during RNN training, we also fixed the maximum sequence length to 40. If the item history sequence is shorter than 40, we fill values with 1, unless stated otherwise in the method description. We always set the likelihood to $\max(\min(1 \quad \varepsilon; \hat{y}); \varepsilon)$, where $\varepsilon = 0{:}05$. If we did not, in the case when, e.g., model output is 0.0, and golden truth is 1 for even one item, the likelihood metric is always reduced to 0 for the whole data set. All methods were implemented in PyTorch (Paszke et al., 2019), except gradient boosting trees, where we used XGBoost library (Chen and Guestrin, 2016).

8.1 Logistic Regression

Logistic regression is a simple but effective machine learning method. In our case, it serves as a baseline machine learning method due to ease of training.

8.2 Feed Forward Neural Network

For a simple feed-forward neural network, we implemented a two-layer network with 4 hidden neurons, a ReLU activation function in between, and a sigmoid activation function on top.

8.3 Half-Life Regression

Half-life regression (HLR) is the Duolingo method described in (Settles and Meeder, 2016). It is similar to logistic regression but imposes exponential decay of the forgetting curve.

It is based on an assumption of probability of recalling an item from memory is

$$p = 2^{-\frac{D}{h}} \tag{1}$$

Where D is lag time (time in days elapsed from the last time the course item was reviewed), h is the half-life, which is the measure of the strength of students' memory of the course item.

Half-life is estimated:

$$\hat{h}_\Theta = 2^{\Theta x}; \tag{2}$$

where x are variables related to student course and item history and Θ are model parameters.

Thus, the estimated probability of recalling a word is:

$$\hat{p}_\Theta = 2^{-\frac{D}{2^{\Theta x}}} \tag{3}$$

During HLR training, the following loss function ℓ is optimized:

$$\ell(hp; D; xi; \Theta) = (p \quad \hat{p}_\Theta)^2 + a\left(\frac{D}{\log_2(p)} \quad \hat{h}_\Theta\right) + \lambda \|\Theta\|_2^2; \tag{4}$$

where a and λ are hyperparameters.

This loss function optimizes not only $\hat{p}_\Theta$, but $\hat{h}_\Theta$ as well. The $\lambda \|\Theta\|_2^2$ is model weights L_2 regularization. Optimizing $\hat{h}_\Theta$ was found to improve loss in the original Duolingo paper, but not on the SuperMemo data set. Finally, we employed the following loss:

$$\ell(hp; D; xi; \Theta) = (p \quad \hat{p}_\Theta)^2 + \lambda \|\Theta\|_2^2 \tag{5}$$

We implemented this method with some slight adjustments for the SuperMemo data set. The main difference is that the SuperMemo data set allows only binary expected values (0 for not remembering in the first attempt in the session, 1 for remembering in the first attempt in the session). This is contrary to Duolingo data set, which allows continuous value based on the attempt number of remembering during the session. Besides, we slightly changed the minimum and maximum boundaries of duration elapsed

[1] See https://supermemo.guru/wiki/Algorithm_SM-17 for the detailed description.

from the last word seen. It means that the word was last seen below 1 day; we set it to 1 day. If the word was last seen above 7 years, we set it to 7 years. This is due to numerical stability because exponential decay assumptions cause floating point overflow in some cases.

8.4 Standard Gradient Boosting Trees

Gradient boosting tree methods usually perform well when it comes down to tabular data. In our experiments, we employed XGboost (Chen and Guestrin, 2016) and used logistic regression as a loss function. The main advantage of this method is its ease of use and good performance out of the box. However, manual search for better than default hyperparameters did not yield significantly better results, so we keep them default.

8.5 Gradient Boosting Trees with Exponential Decay

XGBoost with logistic regression function does not ensure exponential decay assumption. We model the recall probability as

$$\hat{p} = e^{\frac{\mathsf{D}}{o(x)}}; \tag{6}$$

where $o(x)$ is output of XGBoost model. Although, it would be technically difficult to employ this assumption into XGBoost and optimize p directly.

Instead, our approach was to optimize o using the formula:

$$o = \frac{\mathsf{D}}{\ln(p)}: \tag{7}$$

Due to equation 7 indeterminacy, when $p = 0$ or $p = 1$, in practice we employed:

$$o = \frac{\mathsf{D}}{\ln(\min(1 \quad \mathsf{e}; \max(\mathsf{e}; p)))}; \tag{8}$$

where $\mathsf{e} = 0{:}05$.

During inference, recall probability is obtained with equation 6.

This approach does not require XGBoost model architecture modification but only target and prediction transformation. In this method, we used default XGBoost hyperparameters as well.

8.6 Standard RNN

RNN often yields good results in dealing with time series. RNN can take a sequence of any length as input, so its perfect for students learning history.

We implemented 1-layer Long short-term memory (LSTM) ((Hochreiter and Schmidhuber, 1997)) with 256 cell units and trained with MSELoss. During training, we set students learning history to a fixed sequence length of 40 for optimal batching.

8.7 RNN with Exponential Decay

In order to impose exponential decay, we model the probability of item recalling as

$$\hat{p}(x) = e^{\frac{\mathsf{D}}{e^{o(x)}}}; \tag{9}$$

where $o(x)$ is output from the LSTM and D is lag time. Due to the floating point overflow of $e^{o(x)}$, we set the maximum lag time to 3 years instead of 7 years.

For missing values, we set the probability of remembering to 1 and the maximum lag-time, which is 7 years.

9 RESULTS

Due to instability of half-life regression, RNN, and RNN with exponential decay training, we trained the models 10 times and averaged the results.

The results for all described methods are presented in Table 2. The best performing method in the primary metric MacroAvgMAE is RNN with exponential decay, and the best performing method in the secondary metric MacroAvgLikelihood is the feed-forward neural network. Standard XGBoost model surpasses all other methods in not macro averaged metrics, both MAE and Likelihood. In terms of MacroAvgMAE imposing exponential decay helped achieve RNN model better results but worsened XGBoost score. The second best-performing method is SM17, which is an expert algorithm.

10 VERIFICATION ON SYNTHETIC TEST CASES

We prepared a small synthetic data set to verify the results in 8 different cases of user learning history. Each case consists of first student contact with an item and 3 consecutive recalls after some intervals and with relevant grades (*I know*, *Almost*, *Don't know*). After this, we check the probability of recalling an item after 10, 20, 30, 100, 1000 days intervals as returned by a given model; see results in 4.

After a manual inspection on this data set, we concluded:

Table 2: Results of ml and non-ml methods on the SuperMemo.com dataset. Bolded text indicates the best result in the given metric. MacroAvgMAE and MacroAvgLikelihood are primary and secondary metrics.

Method	Likelihood″	MAE #	MacroAvgLikelihood″	**MacroAvgMAE#**
mean from train	0.703±0.027	0.195±0.014	0.500±0.000	0.500±0.000
Leitner system	0.076±0.022	0.526±0.014	0.232±0.032	0.487±0.028
SM-2	0.143±0.040	0.411±0.014	0.269±0.038	0.427±0.025
SM-17	0.614±0.029	0.220±0.012	0.452±0.023	0.390±0.024
logistic regression	0.739±0.024	0.159±0.013	0.526±0.009	0.444±0.010
ff neural network	0.734±0.018	0.206±0.009	**0.539±0.014**	0.435±0.010
half-life regression	0.680±0.036	0.164±0.016	0.492±0.003	0.500±0.005
standard XGBoost	**0.758±0.021**	**0.158±0.010**	0.543±0.011	0.421±0.011
XGBoost with exp decay	0.715±0.025	0.196±0.013	0.509±0.002	0.488±0.003
standard RNN	0.744±0.022	0.163±0.012	0.531±0.009	0.440±0.010
RNN with exp decay	0.527±0.015	0.362±0.012	0.504±0.031	**0.376±0.023**

- XGBoost indicates high recall probability in all cases, even after 3 consecutive recall fails (case 2) and a long interval of 1000 days, which is not in accord with common sense. This model cannot be useful for real-world application, even though it achieved a good MacroAvgMAE result of 0.421, which also leads to the conclusion that automated metrics do not always reflect expectations,
- standard XGBoost learned decay of recalling probability; even though we did not impose it directly into the model,
- standard LSTM didn't learn decay of recalling probability (e.g. case 2) on its own,
- SM17, half-life regression and LSTM models with exponential decay behave as expected.

11 CONCLUSION

Herein, we compared spaced repetition algorithms based on neural networks (LSTMs) with simpler machine learning approaches and existing expert algorithms. In general, methods based on machine learning yielded promising results (with the best result according to our main evaluation metric obtained by an LSTM). Still, some caveats need to be expressed:

- machine-learning methods, including LSTMs, might give good results as measured with an evaluation metric, but still behaving in an impractical manner and breaking natural assumptions (e.g. probability of retention not decreasing even for very long intervals or even higher probabilities of retention for longer intervals),
- this can be alleviated with modifications transplanted from expert methods (e.g. forgetting curve as proposed by Woźniak),
- LSTM is susceptible to large variance and, in practical terms, is more complicated to use than expert methods,
- the ranking of methods depends heavily on the evaluation metric chosen, we claim the evaluation method we called MacroAvgMAE is the most reasonable, but still it is far from obvious how this relates to quality of learning, when a given method is embedded within a real learning application.

One area of improvement for the method based on LSTMs is to equip it with a mechanism to adapt for a specific user/course, just the way expert methods such as SM-17 do.

REFERENCES

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

Graliński, F., Jaworski, R., Borchmann, Ł., and Wierzchoń, P. (2016). Gonito.net – open platform for research competition, cooperation and reproducibility. In Branco, A., Calzolari, N., and Choukri, K., editors, *Proceedings of the 4REAL Workshop: Workshop on Research Results Reproducibility and Resources Citation in Science and Technology of Language*, pages 13–20.

Graliński, F., Wróblewska, A., Stanisławek, T., Grabowski, K., and Górecki, T. (2019). GEval: Tool for debugging NLP datasets and models. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 254–262, Florence, Italy. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.