Noisy Neural Language Modeling for Typing Prediction in BCI Communication

Rui Dong **David A. Smith**
Khoury College of Computer Sciences
Northeastern University
Boston, MA
{dongrui, dasmith}@ccs.neu.edu

Shiran Dudy **Steven Bedrick**
Center for Spoken Language Understanding
Oregon Health & Science University
Portland, OR
{dudy,bedricks}@ohsu.edu

Abstract

Language models have broad adoption in predictive typing tasks. When the typing history contains numerous errors, as in open-vocabulary predictive typing with brain-computer interface (BCI) systems, we observe significant performance degradation in both n-gram and recurrent neural network language models trained on clean text. In evaluations of ranking character predictions, training recurrent LMs on noisy text makes them much more robust to noisy histories, even when the error model is misspecified. We also propose an effective strategy for combining evidence from multiple ambiguous histories of BCI electroencephalogram measurements.

1 Introduction

Brain-computer interface (BCI) systems provide a means of language communication for people who have lost the ability to speak, write, or type, e.g., patients with amyotrophic lateral sclerosis (ALS) or locked-in syndrome (LIS). These systems are designed to detect a user's intent from electroencephalogram (EEG) or other signals and to translate them into typing commands.

Recent studies have shown that incorporating language information into BCI systems can significantly improve both their typing speed and accuracy (Oken et al., 2014; Mora-Cortes et al., 2014; Speier et al., 2016, 2017, 2018; Dudy et al., 2018). Existing methods for optimizing BCI systems with language information either focus on improving the accuracy of symbol classifiers by adding priors from language models(Oken et al., 2014), or on accelerating the typing speed by tying prediction (Dudy et al., 2018), word completion, or automatic error correction (Ghosh and Kristensson, 2017).

Instead of displaying a keyboard layout, these BCI systems present candidate characters sequentially, as in the Shannon game (Shannon, 1951), and then measure users' reactions with EEG or other signals. Predictive performance is thus measured using the mean reciprocal rank of the correct character or the recall of the correct character in the k candidates presented in a batch to the user.

Most previous work on language modeling for BCI employs n-gram language models although the past decade has seen recurrent and other neural architectures surpass these models for many tasks. Furthermore, most predictive typing methods, for BCI or other applications, depend on language models trained on clean text; however, BCI output often contains noise due to misclassification of EEG or other input signals. To the best of our knowledge, language models have rarely been evaluated in with such character-level noise. Recurrent language models, however, could effectively utilize contexts of 200 tokens on average (Khandelwal et al., 2018). Although this might be a disadvantage with noisy histories, we will see that it is no worse than n-gram models with clean training and much better with noisy training.

In addition, existing work mainly focuses on prediction given a single sequence of tokens in the history, but the signal classifier for BCI systems might not always correctly rank the users' intent as the top candidate. Dudy et al. (2018) proposed incorporating ambiguous history into decoding with a joint word-character finite-state model, but typing prediction could not be further improved. Although (Sperber et al., 2017) considered lattice decoding for neural models, the task of integrating multiple candidate histories during online prediction has not been studied.

To address these challenges, we propose to train a noise-tolerant neural language model for online predictive typing and to provide a richer understanding of the effect of the noise on recurrent neural network language models. We aim to answer the following questions: (1) what effect noise in different regions of the history and in different sentence and word positions has on recurrent language model character predictions; (2) how to mitigate performance degradation in LM predictive accuracy with noisy histories; and (3) whether including ambiguous history could help to improve the performance of neural language models.

In this paper, we investigate these questions by training long short-term memory (LSTM: Hochreiter and Schmidhuber, 1997) models on synthetic noisy data generated from the New York Times (NYT) corpus and the SUBTLEXus corpus to cover both formal and colloquial language.

Experimental results show that injecting noise into the training data improves the generalizability of language models on a predictive typing task. Moreover, a neural language model trained on noisy text outperforms n-gram language models trained on noisy or clean text. In fact, some language models trained on clean text do substantially worse than a character unigram baseline when presented with text with only uniform stationary noise. Taking multiple possible candidates into consideration at each time step further improves predictive performance.

2 Related Work

Language Modeling for BCI Systems As noted above, several BCI systems have incorporated n-gram language models trained on clean text (Oken et al., 2014; Speier et al., 2016, 2017). Dudy et al. (2018) propose taking noisy ambiguous outputs from BCI signal classifiers and using a language model to find an optimal path among those output to predict the next letter. Their use of ambiguous histories, however, does not significantly improve performance for a word-character hybrid model.

Noisy Language Models Xie et al. (2017) show that injecting noise into training data for neural network grammar-correction models could achieve comparable performance with parameter regularization and thus make the model more generalizable with limited training data. Belinkov and Bisk (2017) show that machine translation models trained on noisy source text are more robust to the corresponding type of noise. Li et al. (2013) aim at using fixed-history neural network models to analyze the typing stream and predict the next character. Ghosh and Kristensson (2017) describe training a sequence-to-sequence model with attention to automatically correct and complete the current context. Their word-level decoder cannot predict unseen words, as a character or word-character hybrid model could naturally do.

3 Approach

Given an input sequence of characters typed by a user, the goal of typing prediction is to predict the next possible character that the user intends to type, which is the usual objective of language modeling. Both typing speed and typing accuracy could be significantly increased if the user's intended character is ranked higher and presented earlier to the user at each time step. We apply LSTM model, which has been proven effective in capturing long-term dependencies, as our language model.

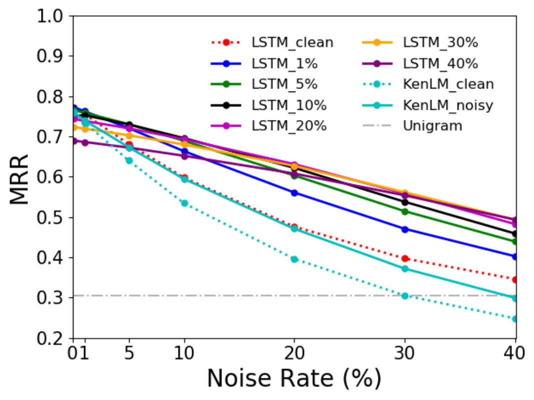
In this paper, we aim to study the effects of noise in the input sequence on the performance of language models for typing prediction in BCI systems. As shown by Belinkov and Bisk (2017) and Xie et al. (2017), the performance of language models and translation models degrades dramatically on text unobserved in the training corpus, but adding appropriate noise to the training corpus could significantly improve accuracy. We therefore propose to generate synthetic noisy data by randomly choosing p percent of characters from both the training and test corpus, and substituting for them a random character excluding the original correct one. Here we use uniform distribution to sample the characters. We then train the language models on the corrupted training set and compare their performance on the corrupted test set.

4 Experiments

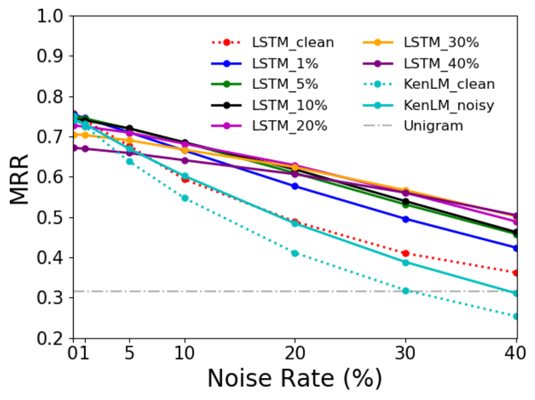
In this section, we first introduce the details of our experimental setup (§4.1). Then we compare the performance of the LSTM and baseline models trained on clean and noisy text (§4.2). Further discussion of the effect of errors on LSTM models follows in §4.3 and §4.4. §4.5 explores whether including multiple candidate histories could further improve predictive performance.

4.1 Experimental Setup

Datasets We evaluate our model on two datasets: the New York Times (NTY) corpus (Sandhaus, 2008) and SUBTLEXus (Brysbaert and New,

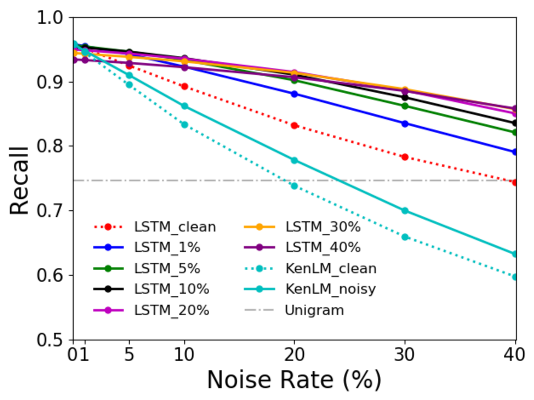


(a) NYT

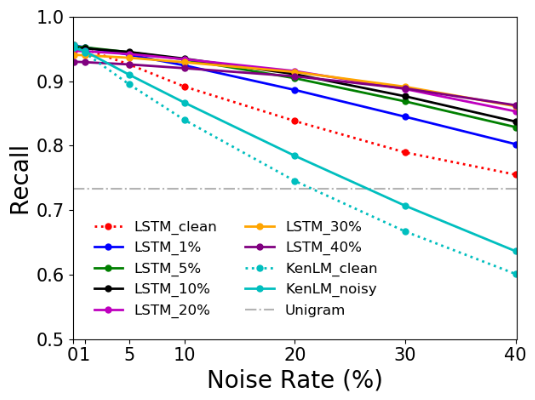


(b) SUBTLEXus

Figure 1: MRR of different models for predictive typing on test sets with different noise rates



(a) NYT



(b) SUBTLEXus

Figure 2: Recall of different models for predictive typing on test sets with different noise rates

2009) corpus of subtitles from movies and television. The NYT has relatively longer sentences, richer vocabulary, and more formal language; SUBTLEXus tends toward colloquial language and shorter sentences. To make a fair comparison between the two corpora, we randomly sample two subsets from them with equal numbers of characters. Both corpora are split into sentences, 80% sentences are randomly sampled as training set while the rest are used as test set. Table 1 summarizes the data.

Dataset	# Sentences	Avg length	Std of Length
NYT	1,750,000	120.11	64.78
SUBTLEXus	5,602,082	37.54	33.42

Table 1: Corpus sentence count and average character length

Baselines and Comparison We compare the LSTM model with two baselines: a character unigram language model (Unigram) and a character n-gram language model with Kneser-Ney smoothing trained with KenLM (Heafield, 2011). Here we set n as 9 and filter all the n-grams appearing less than 5 times. We also compare the LSTM with the OCLM model (Dudy et al., 2018), a joint word-character finite-state model, on the predictive typing task with a ambiguous history. Our LSTM model has 3 layers with 512 hidden units for each layer.

Evaluation Metrics Mean reciprocal rank at 10 (MRR@10) and Recall at 10 (Recall@10) are used for evaluation. Recall@10 reflects whether the correct characters are included in the top 10 candidates suggested by a language model; MRR@10 reveals the rank of the correct character. We use MRR and Recall for short in the following sections. The average values of each metric across all time steps of all sequences are reported.

4.2 Main Results

In this experiment, we compare the LSTM models with Unigram and KenLM models on predictive typing. We first train all the models on clean text to get LSTM_clean, KenLM_clean and Unigram. LSTM and KenLM models are then trained on text with p percent of noise to get the noisy models {LSTM,KenLM}_$p\%$ ($p \in \{1, 5, 10, 20, 30, 40\}$). Since the focus of this paper is on noisy LSTM models, we summarize the performance of all noisy KenLM models as KenLM_noisy: for a given test set, we run KenLM_$p\%$ for each p in $\{1, 5, 10, 20, 30, 40\}$ and choose the best performance as the performance of KenLM_noisy. Then we compare these models on test sets with noise rates varying from 0% (clean) to 40%. Figures 1 and 2 present the performance of all the language models trained on clean or noisy text when evaluating on noisy text.

Noisy or Clean Model? We see that both noisy LSTM models and KenLM models significantly outperform their corresponding clean models on noisy test sets. Although LSTM_clean and KenLM_clean achieve the best performance on clean test data, their performance drops dramatically when applied to noisy data. At 30% noise, KenLM is no better than a unigram baseline; LSTM_clean does not suffer quite as much. The language models trained on clean text are sensitive to the noise rate of the test data, while the language models trained on noisy data performs more robustly, even when only 1% noise is added. Therefore, injecting noise into the training corpus could significantly improve the generalizability of language models on unseen noisy data.

Neural or N-gram Language Model? Figures 1 and 2 show that the accuracy of LSTM models is more stable on unseen noisy text than n-gram models. This advantage, we conjecture, results not only from the LSTMs' incorporation of longer contexts (which might be a disadvantage, but see below), but also because the parameter counts of n-gram models rise with the amount of noise. Even LSTM_clean achieves much higher Recall than KenLM_noisy on test sets with different noise rates. The gap in Recall between them becomes larger when the data is much noisier; however, the difference between their MRR is not that significant. This indicates that errors in the typing history have a more significant impact on the MRR than the Recall of the LSTM_clean model.

How to Choose the Training Noise Rate? The performance of LSTM models trained with different noise rates reveals that all the noisy models works worse on noisier data. But a model performs better when trained and tested on data with matching noise rates. It is unsurprising that LSTM_40% works worst on the test data with 1% of noise, while LSTM_1% works worst on the test set with 40% of noise. When the noise rate of the test set is unknown, training the model with only 10% percent of noise is acceptable for test data with noise rates less than 40%.

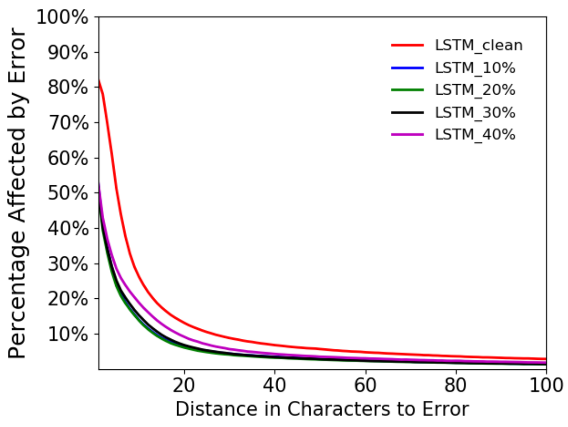


Figure 3: Percentage of predictions whose **MRR** is changed by an error, as a function of distance to the error: LSTM_clean is more sensitive for longer distance.

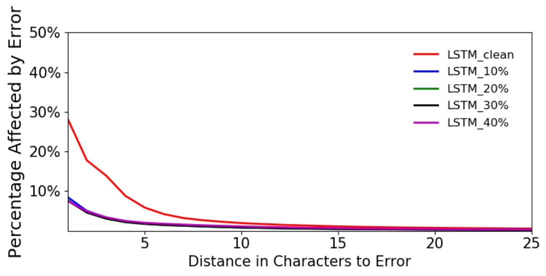
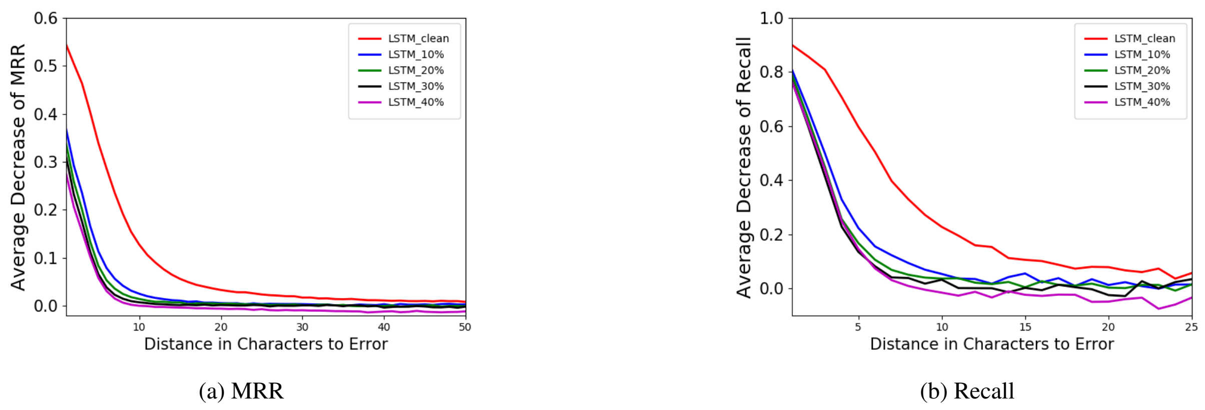


Figure 4: Percentage of predictions whose **recall** is changed by an error, as a function of distance to the error: LSTM_clean is more sensitive for longer distance.

4.3 How Do Errors Affect Nearby and Long-range Predictions?

Since recurrent LMs are sensitive to long-range contexts (Khandelwal et al., 2018), we investigate the impact of errors on the predictions following it. We inject an error into each character of each sentence in turn, and then examine how the LSTM models' predictive performance is affected by the distance to the error. Results are reported on the NYT corpus due to space. SUBTLEXus displays similar behavior.

Percentage of Predictions Affected Figures 3 and 4 show the percentage of predictions affected



(a) MRR (b) Recall

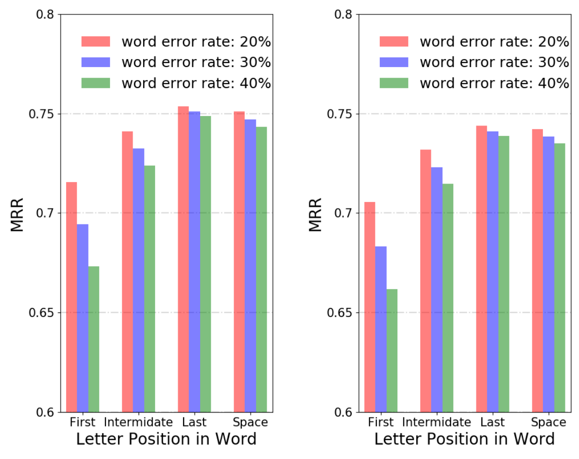
Figure 5: Average decrease of performance at positions with different distances from the error for clean and noisy LSTM models

by the error at positions with different distance to the error: the closer to the error, the higher percentage of predictions affected. The impact of the error is higher and farther on LSTM_clean model than on the noisy models. The noisy models perform similarly, while the most noisy model (LSTM_40%) is slightly more influenced by the error. The error also affects MRR more significantly than Recall. For the noisy LSTMs, the MRR of more than 20% of prediction has been affected by the error within a five-character context, while the Recall of less than 10% of them has been affected. An error also has a longer-range effect on MRR than on Recall. An error affects the MRR of predictions about 40 characters after it, while it only affects the Recall of predictions about five characters after it. While the rank of the correct output among all the characters has been affected by the error, most are still in the top 10.

Performance Degradation Figure 5 presents the average degradation in performance for those predictions affected by the error. Again, we see that the farther away from the error, the smaller the degradation of performance could be observed. The drop of MRR and Recall is less significant about 10 characters after the error for the noisy LSTMs. It can also be observed that the average decrease of MRR and Recall fluctuate around 0 about 10 characters and 5 characters after the error, respectively. This is because the error affects the predictions of the noisy models farther than this distance more randomly and less significantly, i.e., it slightly increases the Recall and MRR of some predictions. This effect is most significant on LSTM_40%, which is trained with the most noisy data. The decrease of recall fluctuates after 5 characters, since the percentage of predictions whose recall is affected is much lower after this distance, as we could see from Figure 4.

4.4 Effect of Error at Different Positions within Words

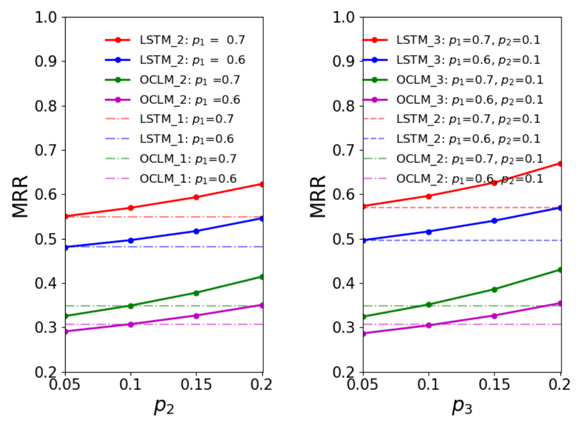
Figure 6 presents the performance of LSTM models when seeing errors in different positions of the words. For each sentence, we randomly choose 20%, 30% and 40% of the words and corrupt each word at different positions: the first letter, the intermediate letters, the last letter, and the spaces after it. Since the character error rate is less than 10%, we test with LSTM_10%. We can see that when the data is noisier, the performance of the model is worse at all positions of the words. Furthermore, the impact of word error rate is most dramatic at the first letter, and the performance of the LSTM gets better when the error moves to later position in the word.



(a) NYT (b) Sublex

Figure 6: MRR of LSTM models tested on words with different noise rates at different positions.

4.5 Exploiting Ambiguous Histories



(a) 2 inputs (b) 3 inputs

Figure 7: Performance of LSTM and OCLM trained with 2 or 3 inputs compared to models trained with 1 less input .

Due to input noise, BCI systems might rank an erroneous character higher than the user's intended one. In this section, we explore whether incorporating ambiguous histories into the model could further improve the performance of neural language models on a simulated predictive typing task.

We simulate ambiguous histories by synthetically generating the top n candidate characters as well as their probabilities predicted by BCI systems at each time step. We ensure that adding one candidate will introduce 5% to 20% percent more correct characters to it. The Dirichlet distribution is used to sample a random multinomial distribution to assign the probabilities to the candidates.

A sum of the embeddings of candidate characters, weighted by those characters' probabilities, is then fed as input to the LSTM at each time step. Summing is much more efficient the OCLM or lattice encoder methods (Sperber et al., 2017). LSTM or OCLM models trained with n inputs are named LSTM_n or OCLM_n ($n \in \{1, 2, 3\}$). The correctness probability of characters in the k^{th} candidate is p_k ($k \in \{1, 2, 3\}$). Performance is reported for LSTM and OCLM trained and tested on datasets with matched error rates.

We compare LSTM and OCLM models on randomly sampled 50K sentences from the test set for NYT corpus. This is because OCLM is slow due to the composition operations between finite state transducers. We find that LSTM performance with ambiguous history on the subset is consistent with its performance on the whole NYT test set as well as its performance on the SUBTLEXus test set reported above.

Figure 7 shows that both LSTM and OCLM work better when more correct characters are added into the inputs. We can see from Figure 7 that the MRR of LSTM has been increased by more than 13% percent when an extra candidate with 20% correctness probability is included; however, adding an extra candidate with 5% correctness probability does not significantly improve the performance of LSTM_2 model, compared to the determinstic model LSTM_1. It even causes a degradation in the performance of the OCLM, which means that the OCLM is more sensitive to the noise in the inputs.

Figure 7 also shows that when adding an input introduces enough correct characters, the performance of the model improves. Figure 7a shows that LSTM models trained with an ambiguous history of 2 inputs (LSTM_2) outperform LSTM models trained with deterministic history (LSTM_1), when the correct probability of the second candidate p_2 is more than 5%. Similar observation could be made for LSTM models with 3 inputs compared with those with 2 inputs in Figure 7b.

We can also see that the LSTM model significantly outperforms the OCLM on ambiguous histories. The OCLM has access to both word and character information, while the LSTM is trained on character sequences alone.

5 Conclusion

In this paper, we propose training language models on noisy text to improve their robustness on unseen noisy text. We also investigate the impact of errors in the history on the performance of recurrent language models. An efficient method of integrating ambiguous histories further improves model performance. We expect to experiment with more realistic error distributions as more real typing data becomes available.

Acknowledgments

We would like to thank the reviewers of the SLPAT workshop for their insightful comments and feedback. This work was supported by NIH grants 5R01DC009834-09 and 1R01DC015999-01. Any views, findings, conclusions, or recommendations expressed do not necessarily reflect those of the NIH.

References

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Marc Brysbaert and Boris New. 2009. Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English. *Behavior Research Methods*, 41(4):977–990.

Shiran Dudy, Shaobin Xu, Steven Bedrick, and David Smith. 2018. A multi-context character prediction model for a brain-computer interface. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, pages 72–77.

Shaona Ghosh and Per Ola Kristensson. 2017. Neural networks for text correction and completion in keyboard decoding. *arXiv preprint arXiv:1709.06429*.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proc. Workshop on Statistical Machine Translation*, pages 187–197.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*.

Jun Li, Karim Ouazzane, Hassan B Kazemian, and Muhammad Sajid Afzal. 2013. Neural network approaches for noisy language modeling. *IEEE Transactions on Neural Networks and Learning Systems*, 24(11):1773–1784.

Anderson Mora-Cortes, Nikolay Manyakov, Nikolay Chumerin, and Marc Van Hulle. 2014. Language model applications to spelling with brain-computer interfaces. *Sensors*, 14(4):5967–5993.

Barry S. Oken, Umut Orhan, Brian Roark, Deniz Erdogmus, Andrew Fowler, Aimee Mooney, Betts Peters, Meghan Miller, and Melanie B. Fried-Oken. 2014. Brain-computer interface with language model–electroencephalography fusion for locked-in syndrome. *Neurorehabilitation and Neural Repair*, 28(4):387–394.

Evan Sandhaus. 2008. The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Claude E. Shannon. 1951. Prediction and entropy of printed English. *Bell System Technical Journal*, 30(1).

William Speier, C. Arnold, and Nader Pouratian. 2016. Integrating language models into classifiers for BCI communication: A review. *Journal of Neural Engineering*, 13(3).

William Speier, Corey Arnold, Nand Chandravadia, Dustin Roberts, Shrita Pendekanti, and Nader Pouratian. 2018. Improving p300 spelling rate using language models and predictive spelling. *Brain-Computer Interfaces*, 5(1):13–22.

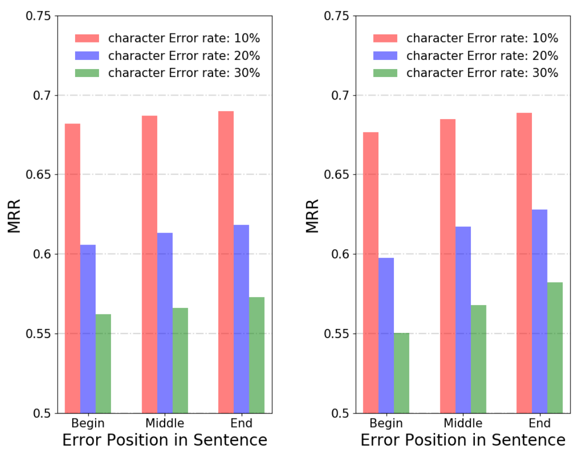
William Speier, Nand Chandravadia, Dustin Roberts, Shrita Pendekanti, and Nader Pouratian. 2017. Online BCI typing using language model classifiers by ALS patients in their homes. *Brain-Computer Interfaces*, 4(1-2):114–121.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. In *EMNLP*.

Ziang Xie, Sida I Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y Ng. 2017. Data noising as smoothing in neural network language models. *arXiv preprint arXiv:1703.02573*.

A Appendices

A.1 Effect of Error at Different Positions within Sentences



(a) NYT (b) Sublex

Figure 8: MRR of LSTM models tested on sentences with different noise rates at different positions.

We also investigate how the position of errors within sentences affects the predictive performance of LSTM models. Figure 8 shows the results of noisy LSTM models tested on sentences where errors appearing in beginning, middle, and end sections of a sentence. Each sentence is split into thirds, and an equal percentage of noise is added to each section in turn. Here, LSTM models are trained and tested on data with consistent noise rate. We can see that the performance of noisy LSTM models does not vary too much when the errors appear in different sections of the sentences. Errors appear at earlier positions in a sentence do not have a significant impact on the predictions in later positions in the sentence.