NeuSpell: A Neural Spelling Correction Toolkit

Sai Muralidhar Jayanthi, Danish Pruthi, Graham Neubig
Language Technologies Institute
Carnegie Mellon University
{sjayanth, ddanish, gneubig}@cs.cmu.edu

Abstract

We introduce NeuSpell, an open-source toolkit for spelling correction in English. Our toolkit comprises ten different models, and benchmarks them on naturally occurring misspellings from multiple sources. We find that many systems do not adequately leverage the context around the misspelt token. To remedy this, (i) we train neural models using spelling errors in context, synthetically constructed by reverse engineering isolated misspellings; and (ii) use contextual representations. By training on our synthetic examples, correction rates improve by 9% (absolute) compared to the case when models are trained on randomly sampled character perturbations. Using richer contextual representations boosts the correction rate by another 3%. Our toolkit enables practitioners to use our proposed and existing spelling correction systems, both via a unified command line, as well as a web interface. Among many potential applications, we demonstrate the utility of our spell-checkers in combating adversarial misspellings. The toolkit can be accessed at neuspell.github.io.[1]

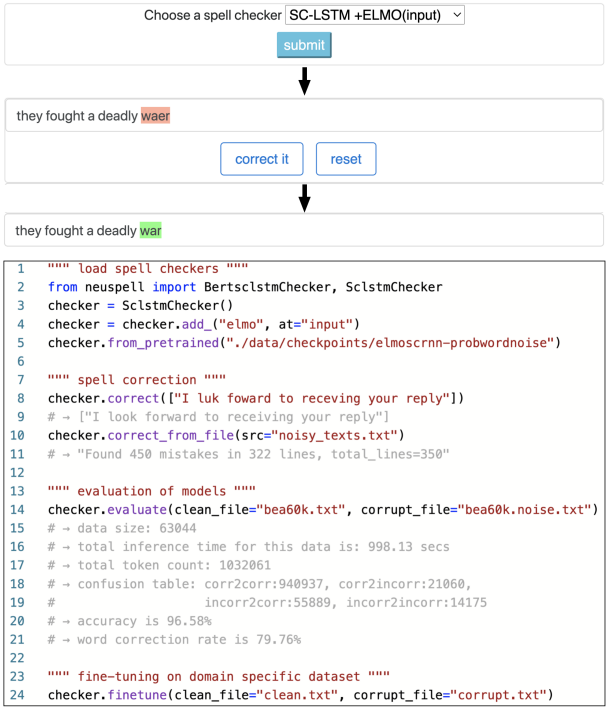

Figure 1: Our toolkit's web and command line interface for spelling correction.

1 Introduction

Spelling mistakes constitute the largest share of errors in written text (Wilbur et al., 2006; Flor and Futagi, 2012). Therefore, spell checkers are ubiquitous, forming an integral part of many applications including search engines, productivity and collaboration tools, messaging platforms, etc. However, many well performing spelling correction systems are developed by corporations, trained on massive proprietary user data. In contrast, many freely available off-the-shelf correctors such as Enchant (Thomas, 2010), GNU Aspell (Atkinson, 2019), and JamSpell (Ozinov, 2019), do not effectively use the context of the misspelled word. For instance, they fail to disambiguate thaught to *taught* or *thought* based on the context: "Who thaught you calculus?" versus "I never thaught I would be awarded the fellowship."

In this paper, we describe our spelling correction toolkit, which comprises of several neural models that accurately capture context around the misspellings. To train our neural spell correctors, we first curate synthetic training data for spelling correction *in context*, using several text noising strategies. These strategies use a lookup table for word-level noising, and a context-based character-level confusion dictionary for character-level noising. To populate this lookup table and confusion matrix, we harvest isolated misspelling-correction pairs from various publicly available sources.

[1]Code and pretrained models are available at: https://github.com/neuspell/neuspell

Proceedings of the 2020 EMNLP (Systems Demonstrations), pages 158–164
November 16-20, 2020.

Further, we investigate effective ways to incorporate contextual information: we experiment with contextual representations from pretrained models such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) and compare their efficacies with existing neural architectural choices (§ 5.1).

Lastly, several recent studies have shown that many state-of-the-art neural models developed for a variety of Natural Language Processing (NLP) tasks easily break in the presence of natural or synthetic spelling errors (Belinkov and Bisk, 2017; Ebrahimi et al., 2017; Pruthi et al., 2019). We determine the usefulness of our toolkit as a countermeasure against character-level adversarial attacks (§ 5.2). We find that our models are better defenses to adversarial attacks than previously proposed spell checkers. We believe that our toolkit would encourage practitioners to incorporate spelling correction systems in other NLP applications.

Model	Correction Rates	Time per sentence (milliseconds)
ASPELL (Atkinson, 2019)	48.7	7.3*
JAMSPELL (Ozinov, 2019)	68.9	**2.6***
CHAR-CNN-LSTM (Kim et al., 2015)	75.8	4.2
SC-LSTM (Sakaguchi et al., 2016)	76.7	**2.8**
CHAR-LSTM-LSTM (Li et al., 2018)	77.3	6.4
BERT (Devlin et al., 2018)	79.1	7.1
SC-LSTM		
+ELMO (input)	**79.8**	15.8
+ELMO (output)	78.5	16.3
+BERT (input)	77.0	6.7
+BERT (output)	76.0	7.2

Table 1: Performance of different correctors in the NeuSpell toolkit on the BEA-60K dataset with real-world spelling mistakes. * indicates evaluation on a CPU (for others we use a GeForce RTX 2080 Ti GPU).

2 Models in NeuSpell

Our toolkit offers ten different spelling correction models, which include: (i) two off-the-shelf non-neural models, (ii) four published neural models for spelling correction, (iii) four of our extensions. The details of first six systems are following:

- GNU Aspell (Atkinson, 2019): It uses a combination of metaphone phonetic algorithm,[2] Ispell's near miss strategy,[3] and a weighted edit distance metric to score candidate words.
- JamSpell (Ozinov, 2019): It uses a variant of the SymSpell algorithm,[4] and a 3-gram language model to prune word-level corrections.
- SC-LSTM (Sakaguchi et al., 2016): It corrects misspelt words using semi-character representations, fed through a bi-LSTM network. The semi-character representations are a concatenation of one-hot embeddings for the (i) first, (ii) last, and (iii) bag of internal characters.
- CHAR-LSTM-LSTM (Li et al., 2018): The model builds word representations by passing its individual characters to a bi-LSTM. These representations are further fed to another bi-LSTM trained to predict the correction.
- CHAR-CNN-LSTM (Kim et al., 2015): Similar to the previous model, this model builds word-level representations from individual characters using a convolutional network.
- BERT (Devlin et al., 2018): The model uses a pre-trained transformer network. We average the sub-word representations to obtain the word representations, which are further fed to a classifier to predict its correction.

To better capture the context around a misspelt token, we extend the SC-LSTM model by augmenting it with deep contextual representations from pre-trained ELMo and BERT. Since the best point to integrate such embeddings might vary by task (Peters et al., 2018), we append them either to semi-character embeddings before feeding them to the biLSTM or to the biLSTM's output. Currently, our toolkit provides four such trained models: ELMo/BERT tied at input/output with a semi-character based bi-LSTM model.

Implementation Details Neural models in NeuSpell are trained by posing spelling correction as a sequence labeling task, where a correct word is marked as itself and a misspelt token is labeled as its correction. Out-of-vocabulary labels are marked as `UNK`. For each word in the input text sequence, models are trained to output a probability distribution over a finite vocabulary using a softmax layer.

We set the hidden size of the bi-LSTM network in all models to 512 and use {50,100,100,100} sized convolution filters with lengths {2,3,4,5} respectively in CNNs. We use a dropout of 0.4 on the bi-LSTM's outputs and train the models using cross-entropy loss. We use the BertAdam[5] optimizer for models with a BERT component and the

[2] http://aspell.net/metaphone/
[3] https://en.wikipedia.org/wiki/Ispell
[4] https://github.com/wolfgarbe/SymSpell
[5] github.com/cedrickchee/pytorch-pretrained-BERT

Adam (Kingma and Ba, 2014) optimizer for the remainder. These optimizers are used with default parameter settings. We use a batch size of 32 examples, and train with a patience of 3 epochs.

During inference, we first replace UNK predictions with their corresponding input words and then evaluate the results. We evaluate models for accuracy (percentage of correct words among all words) and word correction rate (percentage of misspelt tokens corrected). We use AllenNLP[6] and Huggingface[7] libraries to use ELMo and BERT respectively. All neural models in our toolkit are implemented using the Pytorch library (Paszke et al., 2017), and are compatible to run on both CPU and GPU environments. Performance of different models are presented in Table 1.

3 Synthetic Training Datasets

Due to scarcity of available parallel data for spelling correction, we noise sentences to generate misspelt-correct sentence pairs. We use 1.6M sentences from the one billion word benchmark (Chelba et al., 2013) dataset as our clean corpus. Using different noising strategies from existing literature, we noise ∼20% of the tokens in the clean corpus by injecting spelling mistakes in each sentence. Below, we briefly describe these strategies.

RANDOM: Following Sakaguchi et al. (2016), this noising strategy involves four character-level operations: permute, delete, insert and replace. We manipulate only the internal characters of a word. The permute operation jumbles a pair of consecutive characters, delete operation randomly deletes one of the characters, insert operation randomly inserts an alphabet and replace operation swaps a character with a randomly selected alphabet. For every word in the clean corpus, we select one of the four operations with 0.1 probability each. We do not modify words of length three or smaller.

WORD: Inspired from Belinkov and Bisk (2017), we swap a word with its noised counterpart from a pre-built lookup table. We collect 109K misspelt-correct word pairs for 17K popular English words from a variety of public sources.[8]

For every word in the clean corpus, we replace it by a random misspelling (with a probability of 0.3) sampled from all the misspellings associated with that word in the lookup table. Words not present in the lookup table are left as is.

PROB: Recently, Piktus et al. (2019) released a corpus of 20M correct-misspelt word pairs, generated from logs of a search engine.[9] We use this corpus to construct a character-level confusion dictionary where the keys are ⟨character, context⟩ pairs and the values are a list of potential character replacements with their frequencies. This dictionary is subsequently used to sample character-level errors in a given context. We use a context of 3 characters, and backoff to 2, 1, and 0 characters. Notably, due to the large number of unedited characters in the corpus, the most probable replacement will often be the same as the source character.

PROB+WORD: For this strategy, we simply concatenate the training data obtained from both WORD and PROB strategies.

4 Evaluation Benchmarks

Natural misspellings in context Many publicly available spell-checkers correctors evaluate on isolated misspellings (Atkinson, 2019; Mitton; Norvig, 2016). Whereas, we evaluate our systems using misspellings in context, by using publicly available datasets for the task of Grammatical Error Correction (GEC). Since the GEC datasets are annotated for various types of grammatical mistakes, we only sample errors of SPELL type.

Among the GEC datasets in BEA-2019 shared task[10], the Write & Improve (W&I) dataset along with the LOCNESS dataset are a collection of texts in English (mainly essays) written by language learners with varying proficiency levels (Bryant et al., 2019; Granger, 1998). The First Certificate in English (FCE) dataset is another collection of essays in English written by non-native learners taking a language assessment exam (Yannakoudakis et al., 2011) and the Lang-8 dataset is a collection of English texts from Lang-8 online language learning website (Mizumoto et al., 2011; Tajiri et al., 2012). We combine data from these four sources to create the BEA-60K test set with nearly 70K spelling mistakes (6.8% of all tokens) in 63044 sentences.

The JHU FLuency-Extended GUG Corpus (JFLEG) dataset (Napoles et al., 2017) is another

[6] allennlp.org/elmo

[7] huggingface.co/transformers/model_doc/bert.html

[8] https://en.wikipedia.org/, dcs.bbk.ac.uk, norvig.com, corpus.mml.cam.ac.uk/efcamdat

[9] https://github.com/facebookresearch/moe

[10] www.cl.cam.ac.uk/research/nl/bea2019st/

Spelling correction systems in NeuSpell (Word-Level Accuracy / Correction Rate)

	Synthetic		Natural		Ambiguous	
	WORD-TEST	PROB-TEST	BEA-60K	JFLEG	BEA-4660	BEA-322
ASPELL (Atkinson, 2019)	43.6 / 16.9	47.4 / 27.5	68.0 / 48.7	73.1 / 55.6	68.5 / 10.1	61.1 / 18.9
JAMSPELL (Ozinov, 2019)	90.6 / 55.6	93.5 / 68.5	97.2 / 68.9	98.3 / 74.5	**98.5** / 72.9	**96.7** / 52.3
CHAR-CNN-LSTM (Kim et al., 2015)	97.0 / 88.0	96.5 / 84.1	96.2 / 75.8	97.6 / 80.1	97.5 / 82.7	94.5 / 57.3
SC-LSTM (Sakaguchi et al., 2016)	97.6 / 90.5	96.6 / 84.8	96.0 / 76.7	97.6 / 81.1	97.3 / 86.6	94.9 / 65.9
CHAR-LSTM-LSTM (Li et al., 2018)	98.0 / 91.1	97.1 / 86.6	96.5 / 77.3	97.6 / 81.6	97.8 / 84.0	95.4 / 63.2
BERT (Devlin et al., 2018)	**98.9 / 95.3**	**98.2 / 91.5**	93.4 / 79.1	**97.9 / 85.0**	98.4 / **92.5**	96.0 / **72.1**
SC-LSTM						
+ELMO (input)	98.5 / 94.0	97.6 / 89.1	96.5 / **79.8**	97.8 / **85.0**	98.2 / 91.9	96.1 / 69.7
+ELMO (output)	97.9 / 91.4	97.0 / 86.1	**98.0** / 78.5	96.4 / 76.7	97.9 / 88.1	95.2 / 63.2
+BERT (input)	98.7 / 94.3	97.9 / 89.5	96.2 / 77.0	97.8 / 83.9	98.4 / 90.2	96.0 / 67.8
+BERT (output)	98.1 / 92.3	97.2 / 86.9	95.9 / 76.0	97.6 / 81.0	97.8 / 88.1	95.1 / 67.2

Table 2: Performance of different models in NeuSpell on natural, synthetic, and ambiguous test sets. All models are trained using PROB+WORD noising strategy.

collection of essays written by English learners with different first languages. This dataset contains 2K spelling mistakes (6.1% of all tokens) in 1601 sentences. We use the BEA-60K and JFLEG datasets only for the purposes of evaluation, and do not use them in training process.

Synthetic misspellings in context From the two noising strategies described in §3, we additionally create two test sets: WORD-TEST and PROB-TEST. Each of these test sets contain around 1.2M spelling mistakes (19.5% of all tokens) in 273K sentences.

Ambiguous misspellings in context Besides the natural and synthetic test sets, we create a challenge set of ambiguous spelling mistakes, *which require additional context to unambiguously correct them*. For instance, the word whitch can be corrected to "witch" or "which" depending upon the context. Simliarly, for the word begger, both "bigger" or "beggar" can be appropriate corrections. To create this challenge set, we select all such misspellings which are either 1-edit distance away from two (or more) legitimate dictionary words, or have the same phonetic encoding as two (or more) dictionary words. Using these two criteria, we sometimes end up with inflections of the same word, hence we use a stemmer and lemmatizer from the NLTK library to weed those out. Finally, we manually prune down the list to 322 sentences, with one ambiguous mistake per sentence. We refer to this set as BEA-322.

We also create another larger test set where we artificially misspell two different words in sentences to their common ambiguous misspelling. This process results in a set with 4660 misspellings in 4660 sentences, and is thus referred as BEA-4660. Notably, for both these ambiguous test sets, a spelling correction system that doesn't use any context information can at best correct 50% of the mistakes.

5 Results and Discussion

5.1 Spelling Correction

We evaluate the 10 spelling correction systems in NeuSpell across 6 different datasets (see Table 2). Among the spelling correction systems, all the neural models in the toolkit are trained using synthetic training dataset, using the PROB+WORD synthetic data. We use the recommended configurations for Aspell and Jamspell, but do not fine-tune them on our synthetic dataset. In all our experiments, vocabulary of neural models is restricted to the top 100K frequent words of the clean corpus.

We observe that although off-the-shelf checker Jamspell leverages context, it is often inadequate. We see that models comprising of deep contextual representations consistently outperform other existing neural models for the spelling correction task. We also note that the BERT model performs consistently well across all our benchmarks. For the ambiguous BEA-322 test set, we manually evaluated corrections from Grammarly—a professional paid service for assistive writing.[11] We found that our best model for this set, i.e. BERT, outperforms corrections from Grammarly (72.1% vs 71.4%) We attribute the success of our toolkit's well performing models to (i) better representations of the context, from large pre-trained models; (ii) swap invariant semi-character representations; and (iii) training models with synthetic data consisting of noise patterns from real-world misspellings. We follow up these results with an ablation study to understand the role of each noising strategy (Ta-

[11] Retrieved on July 13, 2020 .

Sentiment Analysis (1-char attack / 2-char attack)

Defenses	No Attack	Swap	Drop	Add	Key	All
			Word-Level Models			
SC-LSTM (Pruthi et al., 2019)	79.3	**78.6 / 78.5**	69.1 / 65.3	65.0 / 59.2	69.6 / 65.6	63.2 / 52.4
SC-LSTM+ELMO(input) (F)	**79.6**	77.9 / 77.2	**72.2 / 69.2**	**65.5 / 62.0**	**71.1 / 68.3**	**64.0 / 58.0**
			Char-Level Models			
SC-LSTM (Pruthi et al., 2019)	70.3	65.8 / 62.9	58.3 / 54.2	**54.0 / 44.2**	**58.8** / 52.4	**51.6** / 39.8
SC-LSTM+ELMO(input) (F)	**70.9**	**67.0 / 64.6**	**61.2 / 58.4**	53.0 / 43.0	58.1 / **53.3**	51.5 / **41.0**
			Word+Char Models			
SC-LSTM (Pruthi et al., 2019)	80.1	79.0 / 78.7	69.5 / 65.7	64.0 / **59.0**	66.0 / 62.0	61.5 / **56.5**
SC-LSTM+ELMO(input) (F)	**80.6**	**79.4 / 78.8**	**73.1 / 69.8**	**66.0** / 58.0	**72.2 / 68.7**	**64.0** / 54.5

Table 3: We evaluate spelling correction systems in NeuSpell against adversarial misspellings.

ble 4).[12] For each of the 5 models evaluated, we observe that models trained with PROB noise outperform those trained with WORD or RANDOM noises. Across all the models, we further observe that using PROB+WORD strategy improves correction rates by at least 10% in comparison to RANDOM noising.

Spelling Correction (Word-Level Accuracy / Correction Rate)

Model	Train Noise	Natural test sets BEA-60K	JFLEG
CHAR-CNN-LSTM (Kim et al., 2015)	RANDOM	95.9 / 66.6	97.4 / 69.3
	WORD	95.9 / 70.2	97.4 / 74.5
	PROB	96.1 / 71.4	97.4 / 77.3
	PROB+WORD	96.2 / 75.5	97.4 / 79.2
SC-LSTM (Sakaguchi et al., 2016)	RANDOM	96.1 / 64.2	97.4 / 66.2
	WORD	95.4 / 68.3	97.4 / 73.7
	PROB	95.7 / 71.9	97.2 / 75.9
	PROB+WORD	95.9 / 76.0	97.6 / 80.3
CHAR-LSTM-LSTM (Li et al., 2018)	RANDOM	96.2 / 67.1	97.6 / 70.2
	WORD	96.0 / 69.8	97.5 / 74.6
	PROB	96.3 / 73.5	97.4 / 78.2
	PROB+WORD	96.3 / 76.4	97.5 / 80.2
BERT (Devlin et al., 2018)	RANDOM	**96.9** / 66.3	**98.2** / 74.4
	WORD	95.3 / 61.1	97.3 / 70.4
	PROB	96.2 / 73.8	97.8/ 80.5
	PROB+WORD	96.1 / 77.1	97.8 / 82.4
SC-LSTM +ELMO (input)	RANDOM	**96.9** / 69.1	97.8 / 73.3
	WORD	96.0 / 70.5	97.5 / 75.6
	PROB	96.8 / 77.0	97.7 / 80.9
	PROB+WORD	96.5 / **79.2**	97.8 / **83.2**

Table 4: Evaluation of models on the natural test sets when trained using synthetic datasets curated using different noising strategies.

5.2 Defense against Adversarial Mispellings

Many recent studies have demonstrated the susceptibility of neural models under word- and character-level attacks (Alzantot et al., 2018; Belinkov and Bisk, 2017; Piktus et al., 2019; Pruthi et al., 2019). To combat adversarial misspellings, Pruthi et al. (2019) find spell checkers to be a viable defense. Therefore, we also evaluate spell checkers in our toolkit against adversarial misspellings.

We follow the same experimental setup as Pruthi et al. (2019) for the sentiment classification task under different adversarial attacks. We finetune SC-LSTM+ELMO(input) model on movie reviews data from the Stanford Sentiment Treebank (SST) (Socher et al., 2013), using the same noising strategy as in (Pruthi et al., 2019). As we observe from Table 3, our corrector from NeuSpell toolkit (SC-LSTM+ELMO(input)(F)) outperforms the spelling corrections models proposed in (Pruthi et al., 2019) in most cases.

6 Conclusion

In this paper, we describe NeuSpell, a spelling correction toolkit, comprising ten different models. Unlike popular open-source spell checkers, our models accurately capture the context around the misspelt words. We also supplement models in our toolkit with a unified command line, and a web interface. The toolkit is open-sourced, free for public use, and available at `https://github.com/neuspell/neuspell`. A demo of the trained spelling correction models can be accessed at `https://neuspell.github.io/`.

Acknowledgements

The authors thank Punit Singh Koura for insightful discussions and participation during the initial phase of the project.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*,

[12] To fairly compare across different noise types, in this experiment we include only 50% of samples from each of PROB and WORD noises to construct the PROB+WORD noise set.

pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Kevin Atkinson. 2019. Gnu aspell.

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling.

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

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification.

Michael Flor and Yoko Futagi. 2012. On using context for automatic correction of non-word misspellings in student essays. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 105–115, Montréal, Canada. Association for Computational Linguistics.

Sylviane Granger. 1998. The computerized learner corpus: a versatile new source of data for sla research.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Hao Li, Yang Wang, Xinyu Liu, Zhichao Sheng, and Si Wei. 2018. Spelling error correction using a nested rnn model and pseudo training data.

Roger Mitton. Corpora of misspellings.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234, Valencia, Spain. Association for Computational Linguistics.

Peter Norvig. 2016. Spelling correction system.

Filipp Ozinov. 2019. Jamspell.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Aleksandra Piktus, Necati Bora Edizel, Piotr Bojanowski, Edouard Grave, Rui Ferreira, and Fabrizio Silvestri. 2019. Misspelling oblivious word embeddings. *Proceedings of the 2019 Conference of the North*.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Keisuke Sakaguchi, Kevin Duh, Matt Post, and Benjamin Van Durme. 2016. Robsut wrod reocginiton via semi-character recurrent neural network.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 198–202, Jeju Island, Korea. Association for Computational Linguistics.

Reuben Thomas. 2010. Enchant.

W. John Wilbur, Won Kim, and Natalie Xie. 2006. Spelling correction in the pubmed search engine. *Inf. Retr.*, 9(5):543–564.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational*

Linguistics: Human Language Technologies, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.