Noisy Channel for Low Resource Grammatical Error Correction

Flachs, Simon; Lacroix, Ophélie; Søgaard, Anders

Published in:
Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications

DOI:
10.18653/v1/W19-4420

Publication date:
2019

Citation for published version (APA):
Flachs, S., Lacroix, O., & Søgaard, A. (2019). Noisy Channel for Low Resource Grammatical Error Correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications* (pp. 191-196). Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-4420

Noisy Channel for Low Resource Grammatical Error Correction

Simon Flachs[1,2], **Ophélie Lacroix**[1], **Anders Søgaard**[2]
[1] Siteimprove, Denmark
[2] CoAStaL DIKU, Department of Computer Science, University of Copenhagen, Denmark
{sfl, ola}@siteimprove.com,
soegaard@di.ku.dk

Abstract

This paper describes our contribution to the low-resource track of the BEA 2019 shared task on Grammatical Error Correction (GEC). Our approach to GEC builds on the theory of the noisy channel by combining a channel model and language model. We generate confusion sets from the Wikipedia edit history and use the frequencies of edits to estimate the channel model. Additionally, we use two pre-trained language models: 1) Google's BERT model, which we fine-tune for specific error types and 2) OpenAI's GPT-2 model, utilizing that it can operate with previous sentences as context. Furthermore, we search for the optimal combinations of corrections using beam search.

1 Introduction

Grammatical Error Correction Grammatical Error Correction (GEC) is the task of automatically correcting grammatical errors in written text. The task is relevant for users producing text through text interfaces, both as assistance during the writing process and for proofreading already written work. In recent years, GEC has received increasing attention in the research community with several shared tasks on the topic, such as CoNLL 13-14 (Ng et al., 2013, 2014), HOO (Dale and Kilgarriff, 2011), and AESW (Daudaravicius et al., 2016), and most recently the BEA 2019 shared task on GEC (Bryant et al., 2019), which this work is a contribution to.

Supervised GEC Current state-of-the-art approaches to GEC use a supervised machine translation setup (Ge et al., 2018; Grundkiewicz and Junczys-Dowmunt, 2018), that relies on large amounts of annotated learner data. This means that systems do not generalize well to non-learner domains and that these approaches do not work well for low-resource languages. As most existing datasets are not freely available for commercial use, the supervised approach also limits industrial uses.

Unsupervised GEC In order to combat these problems, in recent years several approaches to GEC have used the concept of language modeling, which allows for training GEC systems without supervised data, and have so far given promising results. Bryant and Briscoe (2018) uses a 5-gram language model while Makarenkov et al. (2019) uses a bidirectional LSTM-based language model. Kaili et al. (2018) fine-tunes LSTM-based language models for specific error types.

Using a language modeling approach means that we can create models that are trained unsupervised by only being based on high quality native text corpora. This means that our systems will only require a small amount of labeled data for tuning purposes. We can therefore build GEC systems for any language given enough native text.

The Noisy Channel The core idea that these language modeling approaches are using for GEC is that low probability sequences are more likely to contain grammatical errors than high probability sequences. However this formulation does not take into account the writer's likelihood of making particular errors. For example, "then" $\rightarrow$ "than" is much more common than "then" $\rightarrow$ "the" due to an underlying similarity in phonetics.

In order to take this into account we utilize the concept of the noisy channel model, which allows for modeling the users likelihood of making particular errors, instead of only relying on which sequences of words are more probable.

Contributions In the following, we present our low-resource approach to GEC, which ranked as the 6th best performing system in the low-resource

Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 191–196
Florence, Italy, August 2, 2019. ©2019 Association for Computational Linguistics

track of the BEA 2019 shared task. We utilize confusion sets and edit statistics gathered from the Wikipedia edit history, as well as unsupervised language models in a noisy channel setting.

Our contributions are 1) formalizing GEC in the noisy channel framework, 2) generating confusion sets from the Wikipedia edit history, 3) estimating a channel model based on frequencies of edits from the confusion sets, 4) combining existing pre-trained language models, with each their own strength, 5) specializing models for specific grammatical error types, and 6) using beam search to find the optimal combination of corrections.

2 The Noisy Channel

The intuition of the noisy channel model (Kernighan et al., 1990; Mays et al., 1990) is that for any given word in a sentence, we have a true underlying word, that has been passed through a noisy communication channel, which potentially has modified the word into an erroneous surface form.

Our goal is to build a model of the channel. With this, given a confusion set, we can pass every candidate correction through this noisy channel to see which one is most likely to have produced the surface word.

The noisy channel model can be formulated as a form of Bayesian inference. Given a potentially erroneous surface word, x, we want to find the hidden word, c^*, from all candidates $c \in C$, that generated x.

$$\hat{c} = \arg\max_{c \in C} P(c|x)$$

Using Bayes' rule this can be restated as

$$\hat{c} = \arg\max_{c \in C} P(x|c) * P(c)$$

where $P(x|c)$ is the likelihood of the noisy channel producing a particular x. This is referred to as the channel model. The prior probability of a hidden word, $P(c)$, is modeled by a language model (Jurafsky and Martin, 2009).

3 System

Our system is a combination of several components: a PoS tagger, the channel model, two language models (BERT and GPT-2) and beam search. We first PoS tag the sentence. Then, the sentence is processed from left to right, and for every word x, we identify the set C of possible correction candidates, based on the PoS tag and our generated confusion sets. We then pick the $c \in C$ with the highest $P(c|x)$ estimated using our components in the following formula:

$$P(c|x) = P_{Channel} * P_{BERT} * P_{GPT-2}$$

We allow the system to consider multiple hypotheses by using beam search, which continuously keeps track of a beam of the most likely hypotheses.

In the following, we describe the different components that make up our GEC system in more detail.

3.1 Channel Model

We estimate the channel model in two ways, depending if the written word is in our vocabulary (real-word error) or not (non-word error).

Real-word errors In order to estimate the channel model $P(x|c)$ for real-word errors, we first make a simplifying assumption that a human only makes a mistake for 1 in 20 words. This means that there is a 5% probability (denoted as α) of the surface word x being wrong. This probability can be distributed between the candidate corrections taken from the confusion set. For a given candidate word c_i we can calculate the channel probability using frequency counts of edits for all candidates in C. We gather frequency counts from the Wikipedia edit history (§ 4.1).

$$P(x|c_i) = \alpha * \frac{|x \rightarrow c_i|}{\sum_{j=1}^{|C|} |x \rightarrow c_j|}$$

Non-word errors For non-word errors we assume that any x not in our vocabulary and not a named entity[1] is an error. Assuming a list of candidate corrections, we use the inverse Levenshtein distance to distribute the error probability between the candidates. Hereby, candidates which are lexically closer to the original word are made more likely.

3.2 Language Models

For language modeling we use a combination of two pre-trained models that have recently given good results: BERT (Devlin et al., 2018) and GPT-2 (Radford et al., 2019).

[1]as estimated by Spacy, `https://spacy.io`

BERT BERT is a Transformer-based (Vaswani et al., 2017) language model pre-trained on a large text corpus. It estimates probabilities by jointly conditioning on both left and right context. We use the pre-trained BERT-Base Uncased model as a starting point for several models, which are each fine-tuned for specific error types on sentences extracted from a Wikipedia dump. We do three types of fine-tuning, using the default hyperparameters of BERT.

- PoS-based fine-tuning, where a word is removed and the model predicts its PoS tag. This is used to classify which word category should be at the position for verb form errors and noun number errors.
- Word-based fine-tuning, where a word is removed and the model predicts the word from a vocabulary of the most common 40.000 words from the Wikipedia dump. This is used to estimate probabilities for words in our confusion sets.
- Comma prediction, where we remove all commas and let the model predict where to insert commas. Any discrepancies between the produced and original sentence is used as comma edits, if the model is more than 95% certain.

GPT-2 GPT-2 is another Transformer-based language model trained on a dataset of 8 million web pages. GPT-2 only looks at the previous context to estimate probabilities. We take advantage of the fact that GPT-2 is trained using previous sentences as context by including the previous sentence when estimating probabilities.

3.3 Beam Search

Since our error correction models make a decision separately for every word, sometimes conflicting corrections can be made, e.g., “the cats is big.” might be corrected to “the cat are big”. Therefore we utilize beam search in order to efficiently explore combinations of corrections in order to find the optimal output sentence. We utilize a beam width of 3.

4 Confusion Sets

The first step in correcting a sentence is to identify the potentially erroneous tokens (or groups of tokens) and determine a set of possible corrections for each. We use several methods for deducing these confusion sets according to different error types.

4.1 Wikipedia Edit History

We utilize the WikEd Error Corpus (Grundkiewicz and Junczys-Dowmunt, 2014) generated from Wikipedia revision histories to create confusion sets. We only retain edits of sentences where only a single word has been changed. We first end up with a list of confused token pairs which includes all types of edits, i.e., semantic or grammatical. We set up a set of rules to filter the edits not adapted to the task (e.g., the semantic replacements), and infrequent ones. We thus remove confusion pairs which define: (i) the replacement of a verb form (e.g., tense/subject–verb agreement errors); (ii) noun number errors; (iii) replacement of numbers or dates; (iv) synonyms and antonyms (using Wordnet[2] (Miller, 1995)); (v) replacement of pronouns with determiners; (vi) insertion/deletion of content words (e.g., nouns) and numbers; (vii) spelling errors.

We end up with a list of 348 edit pairs and their corresponding frequency counts in the WikedEd Error Corpus (ranging from 741 to 60,184 instances). The list includes, for instance, determiner replacements (e.g., “a”→“an”) and frequently confused tokens (e.g. “to”→“too”). It covers most replacement error types but mostly closed-class words replacements such as R:DET or R:PREP.

4.2 Misspelled Words

Given a misspelled word (which we refer as non-word in the channel model) we use the Enchant library[3] to derive a set of suggestions for corrections. It mostly covers the R:SPELL error type but can also include other replacement types (such as content word replacements).

4.3 Specialized Models

For fine-tuned models on specific error types, we define specific rules (mainly based on Part-of-Speech tags) to detect the corresponding tokens and their possible replacements. We use the Spacy[4] library to PoS-tag the sentences.

[2]`https://wordnet.princeton.edu/`
[3]Wrapper for various spell checker engines.
[4]`https://spacy.io/`

Noun number model We detect the nouns by their PoS-tags: NN (singular) and NNS (plural) and use a list of matching singular/plural nouns derived from Wiktionary[5] to suggest a correction. It covers the R:NOUN:NUM and R:NOUN:INFL error types.

Verb forms model We detect all forms of verbs through their PoS-tags and derive a list of potential corrections (i.e., all possible inflections) using the list of English verb inflections from the Unimorph project (Kirov et al., 2016). Here, we mainly cover the R:VERB:FORM and R:VERB:SVA error types but also cases of R:VERB:INFL and R:VERB:TENSE error types.

5 Discussion

5.1 Results

Results on the BEA 2019 shared task test dataset are listed per edit and error type in Table 1. It is evident, that out approach deals with a wide array of error types, but with varying quality. The model performs particularly well on spelling errors, subject–verb agreement errors and inserting missing commas. However, the model performs rather poorly on the replacement of adjectives, adverbs and conjunctions which are based on confusion sets derived from Wikipedia edits suggesting that more filtering would be necessary.

5.2 Ablation analysis

We do an ablation analysis of the different components of our model to see how each part contributes to the performance. The global results are shown in Table 2. Detailed results per error type are shown in Appendix A for all models.

Beam search removing the beam search results in a considerable drop in $F_{0.5}$ by 2.73. This shows that figuring out how to optimally combine multiple local edits is important.

GPT-2 removing GPT-2 results in the largest drop in $F_{0.5}$ score of 5.09. The drop is large for most error types but the ablation is especially damaging on the precision of verb form errors.

BERT dropping BERT results in a 1.11 drop in $F_{0.5}$ score. This indicates that GPT-2 is pulling most of the weight.

[5] https://www.wiktionary.org/

Error type	#	P	R	F$_{0.5}$
M:PUNCT	422	80.10	38.15	65.66
R:ADJ	24	12.50	4.17	8.93
R:ADV	17	33.33	5.88	17.24
R:CONJ	5	2.22	20.00	2.70
R:DET	129	20.48	52.71	23.34
R:MORPH	128	46.15	18.75	35.71
R:NOUN	70	50.00	8.57	25.42
R:NOUN:INFL	19	42.86	31.58	40.00
R:NOUN:NUM	290	43.79	68.31	47.18
R:ORTH	349	10.20	1.43	4.59
R:OTHER	618	20.43	6.15	13.95
R:PART	15	38.89	46.67	40.23
R:PREP	292	39.49	58.56	42.24
R:PRON	50	34.15	56.00	37.04
R:SPELL	321	76.51	75.08	76.22
R:VERB	134	25.00	2.99	10.10
R:VERB:FORM	169	47.96	55.62	49.32
R:VERB:INFL	7	100.00	85.71	96.77
R:VERB:SVA	146	74.39	83.56	76.06
R:VERB:TENSE	160	42.50	10.62	26.56
U:PUNCT	118	34.90	88.14	39.69
All error types	4498	44.52	28.88	40.17

Table 1: Span-level correction results of our system. We do not show results for the error types we do not predict.

Channel model we ablate the channel model by dividing out probabilities by uniform distribution over the candidates instead of using the frequency counts of the confusion sets and reverse Levenshtein distance. It results in a drop in $F_{0.5}$ score by 0.44.

	P	R	F$_{0.5}$
Chan + BERT + GPT	40.29	29.19	37.44
Chan + BERT + beam	37.03	28.98	35.08
Chan + GPT + beam	42.31	29.89	39.06
BERT + GPT + beam	43.50	29.49	39.73
Chan + BERT + GPT + beam	44.52	28.88	40.17

Table 2: Span-level correction results of the ablated models.

6 Conclusions

In this work we have presented our system for the BEA 2019 shared task on Grammatical Error Correction, which ranked as the 6th best in the low resource track.

Our ablation analysis showed that each of the components of our system has a positive effect on the overall performance, meaning that the combination of all of our components leads to the best score.

Future work could explore using more advanced channel models, such as using phonetic features to determine the similarity of words. Furthermore our approach could also be adapted to handle insertions and deletions. Additionally, there are several parameters that could be tuned for better performance, including for example, α, the probability that the channel inserts an error, and the beam width.

Acknowledgements

This project was supported by Siteimprove and the Innovation Fund of Denmark through an industrial PhD grant.

References

Christopher Bryant and Ted Briscoe. 2018. Language model based grammatical error correction without annotated training data. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 247–253, New Orleans, Louisiana. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 Shared Task on Grammatical Error Correction. In *Proceedings of the 14th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics.

Robert Dale and Adam Kilgarriff. 2011. Helping Our Own: The HOO 2011 Pilot Shared Task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, ENLG '11.

Vidas Daudaravicius, Rafael E. Banchs, Elena Volodina, and Courtney Napoles. 2016. A Report on the Automatic Evaluation of Scientific Writing Shared Task. In *Proceedings of BEA 2016*.

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*.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Reaching Human-level Performance in Automatic Grammatical Error Correction: An Empirical Study. *CoRR*, abs/1807.01270.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2014. The WikEd Error Corpus: A Corpus of Corrective Wikipedia Edits and its Application to Grammatical Error Correction. In *Advances in Natural Language Processing – Lecture Notes in Computer Science*, volume 8686, pages 478–490. Springer.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near Human-Level Performance in Grammatical Error Correction with Hybrid Machine Translation. *CoRR*, abs/1804.05945.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing*, 2nd edition. Pearson London.

Zhu Kaili, Chuan Wang, Ruobing Li, Yang Liu, Tianlei Hu, and Hui Lin. 2018. A Simple but Effective Classification Model for Grammatical Error Correction. *CoRR*, abs/1807.00488.

Mark Kernighan, Kenneth Church, and William A. Gale. 1990. A Spelling Correction Program Based on a Noisy Channel Model. In *Proceedings of the 13th conference on Computational linguistics*.

Christo Kirov, John Sylak-Glassman, Roger Que, and David Yarowsky. 2016. Very-large Scale Parsing and Normalization of Wiktionary Morphological Paradigms. In *Proceedings of LREC 2016*.

Victor Makarenkov, Lior Rokach, and Bracha Shapira. 2019. Choosing the Right Word: Using Bidirectional LSTM Tagger for Writing Support Systems. *CoRR*, abs/1901.02490.

Eric Mays, Fred J. Damerau, and Robert Mercer. 1990. Context Based Spelling Correction. *Information Processing Management*, 27:517–522.

George A Miller. 1995. WordNet: a Lexical Database for English. *Communications of the ACM*, 38(11).

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of CoNLL 2014: Shared Task*.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 Shared Task on Grammatical Error Correction. In *Proceedings of CoNLL 2013: Shared Task*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*.

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 *Proceedings of Advances in neural information processing systems (NIPS 2017)*.

A Results per error type

Error type	**#**	**All models**	**C+B+G**	**C+B+beam**	**C+G+beam**	**B+G+beam**
M:PUNCT	422	65.66	65.86	65.54	64.15	65.65
R:ADJ	24	8.93	7.69	8.20	7.81	15.38
R:ADV	17	17.24	12.20	17.24	16.67	13.16
R:CONJ	5	2.70	1.92	2.65	2.65	2.36
R:DET	129	23.34	19.92	23.15	22.24	23.29
R:MORPH	128	35.71	29.48	28.12	31.18	35.09
R:NOUN	70	25.42	23.81	25.21	23.08	23.81
R:NOUN:INFL	19	40.00	38.46	37.31	69.57	46.67
R:NOUN:NUM	290	47.18	43.82	42.59	47.11	46.46
R:ORTH	349	4.59	4.58	4.57	4.61	4.60
R:OTHER	618	13.95	13.30	14.24	13.29	15.07
R:PART	15	40.23	44.12	38.89	33.98	41.67
R:PREP	292	42.24	39.47	41.46	40.46	42.01
R:PRON	50	37.04	34.25	32.22	34.04	35.48
R:SPELL	321	76.22	73.66	75.59	70.85	75.02
R:VERB	134	10.10	9.76	9.35	11.57	10.47
R:VERB:FORM	169	49.32	44.53	17.86	46.58	48.03
R:VERB:INFL	7	96.77	96.77	96.77	96.77	96.77
R:VERB:SVA	146	76.06	72.73	72.66	73.16	74.88
R:VERB:TENSE	160	26.56	26.88	26.61	31.73	30.03
U:PRON	21	0.00	20.00	0.00	18.52	20.00
U:PUNCT	118	39.69	39.13	39.91	39.79	39.91
All types	4498	40.17	37.44	35.08	39.06	39.73

Table 3: Span-level correction results ($F_{0.5}$) for different error types (we do not show results for the error types that we do not predict). **C**: Channel Model, **B**: BERT, **G**: GPT-2.