

# Spelling Correction with Denoising Transformer

**Alex Kuznetsov**

HubSpot, Inc.

Dublin, Ireland

akuznetsov@hubspot.com

**Hector Urdiales**

HubSpot, Inc.

Dublin, Ireland

hector@hubspot.com

## Abstract

We present a novel method of performing spelling correction on short input strings, such as search queries or individual words. At its core lies a procedure for generating artificial typos which closely follow the error patterns manifested by humans. This procedure is used to train the production spelling correction model based on a transformer architecture. This model is currently served in the HubSpot product search. We show that our approach to typo generation is superior to the widespread practice of adding noise, which ignores human patterns. We also demonstrate how our approach may be extended to resource-scarce settings and train spelling correction models for Arabic, Greek, Russian, and Setswana languages, without using any labeled data.

## 1 Introduction

As search engines in web services rely on user-generated input, they are exposed to a significant degree of noise originating from human error. This leads to 10-15% of web search queries being misspelled (Dalianis, 2002; Cucerzan and Brill, 2004), with percentage of misspellings increasing to up to 20% for long-tail queries (Broder et al., 2009), and 26% for academic search engines (Wang et al., 2003). In order to reduce user effort and increase search results recall, spelling correction systems are used. Traditionally such systems use statistical techniques and noisy channel models (Bassil, 2012; Hasan et al., 2015; Eger et al., 2016; Gupta et al., 2019). However, in recent years, a number of promising deep learning approaches were developed, spanning applications from e-commerce (Zhou et al., 2017) to mobile device keyboards (Ghosh and Kristensson, 2017). One downside of deep learning models is their tendency to overcorrect (i.e. correct inputs which should be left as-is) (Movin, 2018; Zhu et al., 2019). This phenomenon

results in the network performing corrections in cases when they are not expected: initially correct or, conversely, niche or completely gibberish queries.

The main application of our work is the HubSpot search, which is used to find contacts, companies, documents, and many other types of items. This means that spelling correction for it has to support inputs in any language, case, containing punctuation, and special characters. Therefore, it is reasonable to treat a search query as a single entity, which can be composed of any set of UTF-8 characters. This frames the task at hand as a query-to-query problem, with a user’s query being an input, and a correction (or the same query in absence of a correction) being the output. Such problem setting naturally leads us to a deep learning implementation of a spelling corrector, in particular using a model with the transformer architecture (Vaswani et al., 2017).

We stress the importance for the model to produce outputs that are identical to its inputs in a low-confidence setting (unfamiliar or niche query, noisy input, etc). This feature allows us to serve query corrections at a very high precision even on queries containing unique and previously unseen tokens, without the overcorrecting behaviour mentioned above.

Additionally we show that, when combined with the ability to generate an infinite number of realistic (i.e. not simply uniformly random) typos in any language which can be mapped to the QWERTY keyboard, this approach allows to train robust spelling corrections systems for any setting, regardless of the volume of labeled data available. We illustrate this by training spelling correctors for Arabic, Greek, Russian, and Setswana languages, without using any misspelling examples in these languages.

## 2 Related Work

Previous research suggested framing spelling correction as a string-to-string translation or a sequence-to-sequence problem (Hasan et al., 2015; Eger et al., 2016; Zhou et al., 2017; Movin, 2018; Wang et al., 2019a; Zhang et al., 2019). In recent years deep learning approaches to spelling correction were actively explored (Sun et al., 2015; Zhou et al., 2017; Ghosh and Kristensson, 2017; Etoori et al., 2018; Li et al., 2018; Movin, 2018), including applications (Zhang et al., 2019; Grundkiewicz et al., 2019) of a transformer architecture (Vaswani et al., 2017). Several works highlight the overcorrection problem, when the model underestimates the self-transformation probability (Sun et al., 2010; Zhu et al., 2019). Lack of sufficient training data (noisy to correct mappings) is another important problem (Ghosh and Kristensson, 2017).

Introduction of artificial noise was previously explored in order to overcome low volume of training data (Hasan et al., 2015; Etoori et al., 2018; Wang et al., 2019b; Choe et al., 2019; Grundkiewicz et al., 2019). However, to the best of our knowledge, our approach is the first to generate character-level noise using statistics derived automatically and purely from human error patterns and which combine typo type frequencies, typo probability given position in a string, and character confusion sets. We avoid using Gaussian distributions and human-engineered heuristics and, instead, derive all statistics exclusively from search logs data. Etoori et al. (2018) derive human error patterns but no detail is provided about deriving patterns beyond error types. Character confusion sets were used before, predominantly in a Chinese language setting (Liu et al., 2013; Chen et al., 2013; Wang et al., 2018, 2019a). Word level confusion sets were studied as well, focusing on grammatical error correction (Wang et al., 2019b; Choe et al., 2019; Grundkiewicz et al., 2019), preposition usage (Rozovskaya and Roth, 2010), and dyslexic errors (Pedler and Mitton, 2010).

Search engine queries may serve as a useful resource for development of spelling correction models (Cucerzan and Brill, 2004; Gao et al., 2010). It is common to use search engine logs in order to collect user-issued query rewrites as labels for training spelling correction systems (Radlinski and Joachims, 2005; Zhang et al., 2006; Hasan et al., 2015; Zhu et al., 2019). Some researchers (Gao et al., 2010; Sun et al., 2010; Movin, 2018) find

clicks on query suggestions to be another reliable source of labels.

As an alternative to custom deep learning spelling correction models, statistical open-source models may be used, such as SymSpell<sup>1</sup>. We evaluated symspellpy<sup>2</sup> and spelling corrector by Peter Norvig<sup>3</sup> as examples of such models. When tested on a dataset of HubSpot search logs, we found the following disadvantages, with some of them highlighted by authors of these models: 1) on noisy domains like search logs such models overcorrect at the very high rate (i.e. gibberish or incomplete search queries tend to get corrected towards known vocabulary words when it is not desired), 2) absence of a confidence score for model outputs (it is possible to use proxies such as edit distance and token popularity, however such proxies are too coarse), 3) such models can not be trained or tuned. We, therefore, focus on a deep learning approach to address these shortcomings.

## 3 Generating Realistic Typos

Training deep learning spelling correction models is typically based on a large corpus of `<typo_string, correct_string>` pairs. Such pairs can either be collected (Hasan et al., 2015; Movin, 2018) or generated artificially (Felice and Yuan, 2014; Rei et al., 2017). In our case we are constrained by a dataset of 195K unique `<typo_string, correct_string>` pairs which is insufficient for training a sequence-to-sequence model which is capable of generalising to the diversity of data seen at inference time. As our experiments have shown, a model trained on such a small dataset will suffer from the overcorrection behaviour highlighted in earlier studies (Sun et al., 2010; Zhu et al., 2019).

We address these challenges by reverse engineering the way humans make typos, and using this knowledge to generate as many typos as needed on our unlabeled dataset. This dataset is then used to train a denoising autoencoder which learns to attempt corrections only on misspellings of commonly used words, ignoring unfamiliar queries (which can be typos, gibberish or valid long-tail searches).

There are three main parts to the construction of

<sup>1</sup><https://github.com/wolfgarbe/SymSpell>

<sup>2</sup><https://github.com/mammothb/symspellpy>

<sup>3</sup><https://norvig.com/spell-correct.html>

a training dataset: typo mining (§3.1), typo stats extraction (§3.2), and typo generation (§3.3).

### 3.1 Typo Mining

We use unique search queries issued in HubSpot product to look for pairs of similar queries which were fired by the same user close to each other in time (we use a rolling window of 10 subsequent queries). Such pairs often are user-issued query rewrites which may happen due to a spelling mistake. In order to maximise the chance that one query qualifies as a typo of another query, the following set of rules is applied:

- There is a small edit distance between two queries. We allow for maximum Damerau-Levenshtein edit distance (Damerau, 1964) of 1.
- There is a significant (at least 15X) difference in popularity of two queries (i.e. we assume the correct query is much more popular).
- The query is considered to be “correct” if all its tokens are either present in the verified vocabulary (list of known names and English words), or belong to 1.5K most popular tokens in search logs.
- The candidate “typo” query is not composed solely of known tokens (this excludes cases of alternative name spellings).
- Queries do not contain any forbidden special characters (e.g. @, \_, #, \).
- The candidate typo query is not a prefix of a correct query (e.g. excluding pairs such as <jac, jack>, <jess, jessica>, <mick, mickey>).
- Correct query is not a part of a candidate typo query (e.g. excluding pairs such as <jimmy, jim>, <alex, lex>, <anastasia, stas>).

Applying these filters on 94M search queries containing 135M tokens (19M unique) produces a collection of 195K `<typo_string, correct_string>` pairs composed of 296K tokens (210K unique).

### 3.2 Typo Stats Extraction

Using the `<typo_string, correct_string>` pairs from the previous step we extract typo-related statistics. These include: typo type (insertion, substitution, deletion, transposition) frequencies, character confusion matrix (i.e. probability of an accidental swap of any two characters), and distribution of normalised typo positions in a string. This information describes some of the main patterns of how humans make typographical errors, effectively taking into account keyboard and, in part, phonetic proximity.

**Types of typos.** We follow the commonly used convention of classifying string edits into four categories: insertion, substitution, deletion, and transposition. These categories account for 80-95% of all typos (Martins and Silva, 2004). Number of typos in each category for our dataset is presented in Table 1.

Typo Type	Number of Pairs	% of Total
Insertion	64060	32.74
Substitution	75922	38.80
Deletion	34580	17.67
Transposition	21103	10.79
Total	195665	100.00

Table 1: Volume of examples for each typo type.

**Character confusion set.** We find that `<typo_string, correct_string>` pairs which belong to a *Substitution* category are a reliable source of information behind character-level errors. In particular, we are able to derive, for each character, a probability distribution (over all other characters) of making an erroneous substitution. These distributions highlight that keyboard proximity and phonetics are significant drivers of typing errors. We illustrate these findings in Figure 1 with a character confusion set for lower case English alphabet with all other characters excluded for visualisation purposes. Full confusion set contains 75 characters misspelled for 208 characters.

**Position of a typo in a string.** The probability of making a spelling mistake is a function of its normalised position within a string.

We start by finding the first position at which the correct query and the typo query differ, and divide this position by the length of the correct query.

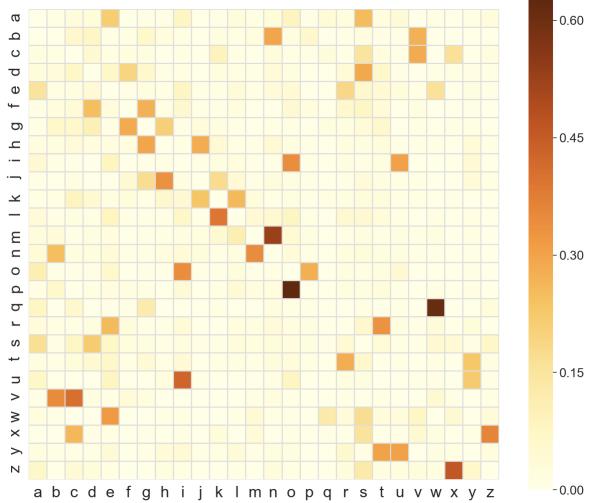


Figure 1: Character confusion set. Restricted to lower case English alphabet for visualisation. Character pairs which are close to each other on a keyboard tend to have higher values. Values in each row sum up to 1.

Normalising typo positions for all `<typo_string, correct_string>` pairs allows us to compute a probability of making a typing mistake by each of 100 percentiles (e.g. probability at 66th percentile corresponds to a probability of making a typo in first 2/3 of the string). Based on an input string length we convert probabilities for 100 percentiles to a probability mass function over all character positions in a string. With typo probabilities assigned to each individual character in an input string, it is trivial to iterate over such string and generate typos, following the patterns exhibited by humans.

We find that typos tend to happen closer to the end of the string, confirming findings in earlier studies (Mitton, 1996). The distribution of normalised typo positions is presented in Figure 2.

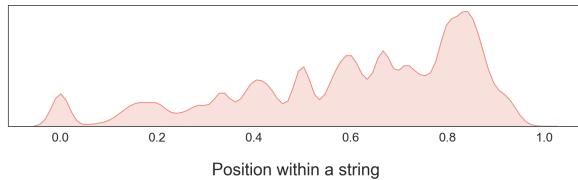


Figure 2: Distribution of a normalised typo position within a string.

### 3.3 Typo Generation

Using the statistics described above, we are able to generate any number of realistic typos that closely mimic human errors. Our algorithm accepts any string as an input and generates a realistic-looking

typo, based on string length and statistics described above. We run this algorithm directly on search logs, attempting typo generation for each record. On average we introduce 1 typo per record, but due to the stochastic nature of the typo generation procedure, some records may have 0 or 2 typos.

Using this procedure we generate a dataset of 94M examples, which is significantly larger than the labeled records we could obtain with any other method. A byte pair encoding (BPE) tokenizer (Sennrich et al., 2015) is then fit on this dataset, resulting in a vocabulary of 12K tokens. Similar to Zhou et al. (2017), we find subword tokenization to be the best way of representing model inputs. The whole dataset is tokenized using the BPE tokenizer, shuffled, and split in 100:1:1 proportions into train, validation, and test sets. This approach allows us to have a training set that is very similar to the data seen in production, thereby minimising possible distribution drifts.

One side effect of constructing `<typo_string, correct_string>` pairs directly from search logs is that noise is introduced over already erroneous queries, like gibberish, niche or partial searches. This may appear detrimental to model performance, however, surprisingly, we observe that in practice noise introduced over erroneous queries does not hurt the quality of model outputs. Instead, given the large size and diversity of the dataset, it forces the model to output sequences identical to the input sequence by default, and to only attempt correction in cases of high certainty (e.g. if a typo is made in a sufficiently popular token). By forcing model outputs to be identical to model inputs in case of gibberish queries, this setup effectively addresses the overcorrection problem (Movin, 2018; Zhu et al., 2019).

Additionally, as some input tokens are compounds (e.g. email addresses, containing first and last names, domain address, etc), this setup forces the model to handle multiple typos in several distinct entities within a single contiguous string.

The ability to train a well performing model directly on the noise generated over unprocessed search logs is surprising, and to the best of our knowledge was not demonstrated before.

## 4 Spelling Correction Model

We train a denoising autoencoder transformer model to recover the original query (without noise)

from the query which potentially contains a typo. As model inputs and labels are generated directly from logs data, the distribution of queries is similar between training and inference settings, thereby minimising distribution drifts and biases. Additionally, as queries are seen by the model according to their popularity, the model will naturally learn most frequent terms and will be forced to learn to ignore (and not correct typos on) infrequent, often erroneous and incomplete queries. The production version of our model is a transformer with 4 layers, 2 attention heads, hidden layer size of 256, trained for 1.6M steps with default learning rate warm-up and decay. This results in a model with 10M trainable parameters. For model implementation we rely on a `tensor2tensor`<sup>4</sup> library (Vaswani et al., 2018) and use the hyper-parameter set defined in the library as `transformer_small`.

## 5 Experiments

Below we present two experiments: one comparing approaches to artificial noise generation, and another demonstrating ability to perform transfer to other languages for which no labels are available. Maximising model quality was not our goal in these experiments, and we expect that additional tuning of the model, vocabulary generation and training procedures, as well as using beam search score as a confidence threshold will yield significant improvements in quality of spelling correction.

### 5.1 Realistic vs Uniformly Generated Typos

We compare two approaches of generating training data: using realistic typos (*Real*) and using a baseline (*Base*) approach which generates typos in a uniformly random way. For *Real* approach we use typo statistics derived from search logs and for *Base* approach all typo types and string positions are treated as equally probable, and characters for *Insertion* and *Substitution* are chosen uniformly at random from a set of lowercase and uppercase English alphabet letters. For this comparison we train identical denoising transformer models on artificially generated typos for two datasets: HubSpot search logs (94M original search queries, not tokenized) and a dataset of 3036 Project Gutenberg books (tokenized into 51M tokens, 989K unique) (Lahiri, 2014). For each dataset we generate both uniform and realistic versions of

a typo for exactly the same set of input strings. Apart from tokenization for the Gutenberg dataset, no data preprocessing is performed. Models trained on the Gutenberg dataset are evaluated on ground truth datasets of English typos: Wikipedia Common Misspellings<sup>5</sup>, and Aspell, Birkbeck, Holbrook datasets<sup>6</sup>. Models trained on HubSpot search logs are evaluated on a dataset of 195K `<typo_string, correct_string>` pairs described in section §3.1. All models are identical to the one described in section §4 and are trained for 200K steps (5-6 hours on Nvidia V100 GPU). We report sequence-level accuracy on both `<typo_string, correct_string>` (*Typos*) and `<correct_string, correct_string>` (*Identity*) pairs. Accuracy on *Identity* pairs is equivalent to  $1 - FPR$ , where *FPR* is False Positive Rate<sup>7</sup>. Results of this experiment are presented in Table 2.

Dataset	Typos		Identity	
	Real	Base	Real	Base
Search Typos	<b>56.84</b>	43.70	96.09	96.83
Wikipedia	<b>65.92</b>	63.58	84.90	86.39
Aspell	<b>40.30</b>	37.66	83.78	84.22
Birkbeck	<b>33.34</b>	29.27	85.14	85.40
Holbrook	<b>17.92</b>	17.25	73.92	74.92

Table 2: Comparison of realistic and uniformly random typo generation approaches.

Experiment results suggest that there is a considerable benefit in generating typos in a realistic manner, which is especially evident in the case of our in-house search typos dataset, from which human error patterns were derived. The fact that error patterns derived from search typos may be successfully transferred to other domains (like Wikipedia, Aspell, and Birkbeck datasets) shows that we are able to at least partially capture fundamental (and not dataset-specific) statistics about human spelling mistakes. In the next section we challenge this conclusion further, attempting to apply our method in non-English domains where no labeled data is available.

<sup>5</sup>[https://en.wikipedia.org/wiki/Wikipedia:Lists\\_of\\_common\\_misspellings\\_for\\_machines](https://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings_for_machines)

<sup>6</sup><https://www.dcs.bbk.ac.uk/~ROGER/corpora.html>

<sup>7</sup>[https://en.wikipedia.org/wiki/False\\_positive\\_rate](https://en.wikipedia.org/wiki/False_positive_rate)

<sup>4</sup><https://github.com/tensorflow/tensor2tensor>

## 5.2 Transfer to Resource-Scarce Languages

Our procedure of training data generation is based on introduction of noise to natural language and relies solely on pre-computed typo-related statistics. Under a bold assumption that such statistics are largely language-agnostic, we show that it is possible to train a denoising transformer-based spelling correction model in settings where no `<typo_string, correct_string>` pairs are available. Leaving other statistics unchanged, we convert the character confusion matrix from English language to a target language using a QWERTY keyboard mapping. This way each English character is mapped to a character on the same keyboard key used in a target language layout. Using updated statistics, we train simple models for Russian<sup>8</sup>, Arabic (Aly and Atiya, 2013), Greek<sup>9</sup>, and Setswana<sup>10</sup> languages. Datasets for Arabic, Greek, and Setswana are split into individual tokens. Datasets for Greek and Russian are lowercased. We use the same model configuration and training procedure as in section §5.1. In Table 3 we report the number of unique examples and tokens for each dataset, alongside with sequence-level accuracy on a test set (not seen by BPE tokenizer and the model during training).

Dataset	Example #	Token #	Accuracy
Arabic	4,096,407	318,521	83.33
Greek	9,491,753	270,269	93.97
Russian	2,679,222	324,867	91.83
Setswana	2,348,161	61,382	94.48

Table 3: Language transfer results.

Results indicate that this simple approach proves itself useful for bootstrapping spelling correction systems in settings when no labels are available. These findings may be especially helpful for languages suffering from scarcity of available resources, such as the majority of languages in Africa (Martinus and Abbott, 2019).

<sup>8</sup>[https://github.com/Koziev/NLP\\_Datasets/blob/master/Samples/prep%2Bnoun.zip](https://github.com/Koziev/NLP_Datasets/blob/master/Samples/prep%2Bnoun.zip)

<sup>9</sup><https://repositori.upf.edu/handle/10230/19963>

<sup>10</sup><https://repo.sadilar.org/handle/20.500.12185/404>

## 6 Production Usage

Trained model is loaded in memory using the TensorFlow (Abadi et al., 2016) SavedModel format, and is fed all input strings shorter than `MAX_INPUT_LENGTH=20`. We limit max input size in order to ignore abnormally long inputs and to provide latency guarantees, as transformer time complexity is quadratic in the input sequence length.

Beam search of size 2 is performed when selecting top output sequence candidate, and we find that increasing beam size gives only minimal quality improvements at the expense of significantly higher latency. Beam search score is treated as a confidence score and is assigned to every prediction. Empirically chosen cut-off of 0.5 is used for serving predictions (i.e. all spelling corrections with score below this threshold are ignored), resulting in 1.5% of queries being corrected. Relationship between confidence threshold and spelling correction rate on HubSpot search logs is presented in Figure 3.

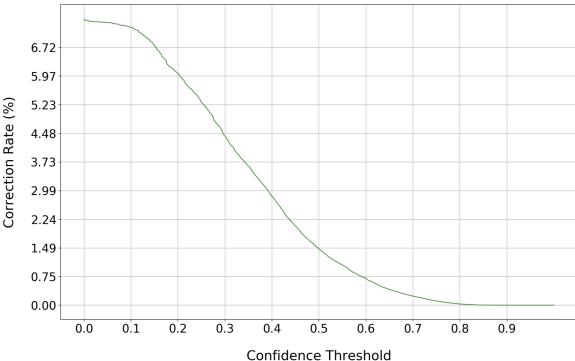


Figure 3: Correction Rate vs Confidence Threshold.

## 7 Future Work

This paper presents our first iteration on deep learning spelling correction, with multiple avenues for further improvement and research. In particular, we leave several items for future work:

- **Model architecture improvements.** Currently, we use a default transformer implementation, and there may be benefit in increasing model capacity, vocabulary and beam search size, using custom architectures, as well as a combination of models suggested by Li et al. (2018). Additional techniques like label smoothing, checkpoint averaging,

and pretraining on a larger corpus may also improve model performance.

- **Personalised recommendations.** Taking user context into account is key to providing a personalised search experience. Our current model is global and ignores user preferences. Embedding user context and using it as a feature may be an appropriate solution for this problem. Model architecture and findings from Gmail Smart Compose (Chen et al., 2019) may be applicable here.
- **Smarter noise generation.** Our current approach to typo generation is better than random but is still far from being perfect at emulating human behavior. For instance, *Insertion* errors depend on both previous and next (relative to the injected character) characters. This is currently not taken into account. Additionally, we have very limited knowledge on how the probability of making a typo changes with the length of the string. Although known to be challenging, generative adversarial models for text (Fedus et al., 2018) may be used in order to generate errors indistinguishable from those of humans.

## 8 Conclusion

We presented a novel method for spelling correction - a denoising autoencoder transformer based on a noise generation procedure which generates artificial spelling mistakes in a realistic manner. Our contributions are three-fold, we: 1) demonstrated that a realistic typo generation procedure is superior to adding noise in a uniform way, 2) presented a way to train a spelling correction model in resource-scarce settings where no labeled data is available, and 3) by using unprocessed search logs showed that training a model directly on data from the target domain is possible and prevents the model from overcorrecting.

## References

- Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283.
- Mohamed Aly and Amir Atiya. 2013. LABR: A large scale Arabic book reviews dataset. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 494–498, Sofia, Bulgaria. Association for Computational Linguistics.
- Youssef Bassil. 2012. Parallel spell-checking algorithm based on yahoo! n-grams dataset. *arXiv preprint arXiv:1204.0184*.
- Andrei Broder, Peter Ciccolo, Evgeniy Gabrilovich, Vanja Josifovski, Donald Metzler, Lance Riedel, and Jeffrey Yuan. 2009. Online expansion of rare queries for sponsored search. In *Proceedings of the 18th international conference on World wide web*, pages 511–520. ACM.
- Kuan-Yu Chen, Hung-Shin Lee, Chung-Han Lee, Hsin-Min Wang, and Hsin-Hsi Chen. 2013. A study of language modeling for chinese spelling check. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 79–83.
- Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al. 2019. Gmail smart compose: Real-time assisted writing. *arXiv preprint arXiv:1906.00080*.
- Yo Joong Choe, Jiyeon Ham, Kyubyong Park, and Yeoil Yoon. 2019. A neural grammatical error correction system built on better pre-training and sequential transfer learning. *arXiv preprint arXiv:1907.01256*.
- Silviu Cucerzan and Eric Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 293–300.
- Hercules Dalianis. 2002. Evaluating a spelling support in a search engine. In *International Conference on Application of Natural Language to Information Systems*, pages 183–190. Springer.
- Fred J Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176.
- Steffen Eger, Tim vor der Brück, and Alexander Mehler. 2016. A comparison of four character-level string-to-string translation models for (ocr) spelling error correction. *The Prague Bulletin of Mathematical Linguistics*, 105(1):77–99.
- Pravallika Etoori, Manoj Chinnakota, and Radhika Mamidi. 2018. Automatic spelling correction for resource-scarce languages using deep learning. In *Proceedings of ACL 2018, Student Research Workshop*, pages 146–152.
- William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: better text generation via filling in the\_. *arXiv preprint arXiv:1801.07736*.

- Mariano Felice and Zheng Yuan. 2014. Generating artificial errors for grammatical error correction. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 116–126.
- Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 358–366. Association for Computational Linguistics.
- Shaona Ghosh and Per Ola Kristensson. 2017. Neural networks for text correction and completion in keyboard decoding. *arXiv preprint arXiv:1709.06429*.
- Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263.
- Jai Gupta, Zhen Qin, Michael Bendersky, and Donald Metzler. 2019. Personalized online spell correction for personal search. In *The World Wide Web Conference*, pages 2785–2791. ACM.
- Saša Hasan, Carmen Heger, and Saab Mansour. 2015. Spelling correction of user search queries through statistical machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 451–460.
- Shibamouli Lahiri. 2014. Complexity of Word Collocation Networks: A Preliminary Structural Analysis. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 96–105, Gothenburg, Sweden. Association for Computational Linguistics.
- Chen Li, Junpei Zhou, Zuyi Bao, Hengyou Liu, Guangwei Xu, and Linlin Li. 2018. A hybrid system for Chinese grammatical error diagnosis and correction. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 60–69, Melbourne, Australia. Association for Computational Linguistics.
- Xiaodong Liu, Kevin Cheng, Yanyan Luo, Kevin Duh, and Yuji Matsumoto. 2013. A hybrid chinese spelling correction using language model and statistical machine translation with reranking. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 54–58.
- Bruno Martins and Mário J Silva. 2004. Spelling correction for search engine queries. In *International Conference on Natural Language Processing (in Spain)*, pages 372–383. Springer.
- Laura Martinus and Jade Z Abbott. 2019. A focus on neural machine translation for african languages. *arXiv preprint arXiv:1906.05685*.
- R. Mitton. 1996. *English spelling and the computer*. Longman Group.
- Maria Movin. 2018. Spelling correction in a music entity search engine by learning from historical search queries.
- Jennifer Pedler and Roger Mitton. 2010. A large list of confusion sets for spellchecking assessed against a corpus of real-word errors. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10)*.
- Filip Radlinski and Thorsten Joachims. 2005. Query chains: learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248. ACM.
- Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial error generation with machine translation and syntactic patterns. *arXiv preprint arXiv:1707.05236*.
- Alla Rozovskaya and Dan Roth. 2010. Generating confusion sets for context-sensitive error correction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 961–970. Association for Computational Linguistics.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.
- Chengjie Sun, Xiaoqiang Jin, Lei Lin, Yuming Zhao, and Xiaolong Wang. 2015. Convolutional neural networks for correcting english article errors. In *Natural Language Processing and Chinese Computing*, pages 102–110. Springer.
- Xu Sun, Jianfeng Gao, Daniel Micol, and Chris Quirk. 2010. Learning phrase-based spelling error models from clickthrough data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 266–274. Association for Computational Linguistics.
- Ashish Vaswani, Samy Bengio, Eugene Brevdo, François Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.
- Dingmin Wang, Yan Song, Jing Li, Jialong Han, and Haisong Zhang. 2018. A hybrid approach to automatic corpus generation for chinese spelling check. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2517–2527.

Dingmin Wang, Yi Tay, and Li Zhong. 2019a. Confusionset-guided pointer networks for chinese spelling check. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5780–5785.

Liang Wang, Wei Zhao, Ruoyu Jia, Sujian Li, and Jingming Liu. 2019b. Denoising based sequence-to-sequence pre-training for text generation. *arXiv preprint arXiv:1908.08206*.

Peiling Wang, Michael W Berry, and Yiheng Yang. 2003. Mining longitudinal web queries: Trends and patterns. *Journal of the american Society for Information Science and technology*, 54(8):743–758.

Shiliang Zhang, Ming Lei, and Zhijie Yan. 2019. Investigation of transformer based spelling correction model for ctc-based end-to-end mandarin speech recognition. *Proc. Interspeech 2019*, pages 2180–2184.

Yang Zhang, Pilian He, Wei Xiang, and Mu Li. 2006. Discriminative reranking for spelling correction. In *Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation*, pages 64–71.

Yingbo Zhou, Utkarsh Porwal, and Roberto Konow. 2017. Spelling correction as a foreign language. *arXiv preprint arXiv:1705.07371*.

Canxiang Zhu, Zhiming Chen, Yang Liu, Juan Hu, Shujuan Sun, Bixiao Cheng, Zhendong, and Xiaoxian Yang. 2019. Automatic query correction for poi retrieval using deep and statistical collaborative model.