Spelling Correction with Denoising Transformer

Alex Kuznetsov
HubSpot, Inc.
Dublin, Ireland
akuznetsov@hubspot.com

Hector Urdiales
HubSpot, Inc.
Dublin, Ireland
hector@hubspot.com

arXiv:2105.05977v1 [cs.CL] 12 May 2021

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[1]. We evaluated symspellpy[2] and spelling corrector by Peter Norvig[3] 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

[1] https://github.com/wolfgarbe/SymSpell

[2] https://github.com/mammothb/symspellpy

[3] 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.