Contextual Multilingual Spellchecker for User Queries

Sanat Sharma
sanatsha@adobe.com
Adobe Inc.
USA

Josep Valls-Vargas
jvallsvargas@adobe.com
Adobe Inc.
USA

Tracy Holloway King
tking@adobe.com
Adobe Inc.
USA

Francois Guerin
guerin@adobe.com
Adobe Inc.
USA

Chirag Arora
charora@adobe.com
Adobe Inc.
USA

ABSTRACT

Spellchecking is one of the most fundamental and widely used search features. Correcting incorrectly spelled user queries not only enhances the user experience but is expected by the user. However, most widely available spellchecking solutions are either lower accuracy than state-of-the-art solutions or too slow to be used for search use cases where latency is a key requirement. Furthermore, most innovative recent architectures focus on English and are not trained in a multilingual fashion and are trained for spell correction in longer text, which is a different paradigm from spell correction for user queries, where context is sparse (most queries are 1–2 words long). Finally, since most enterprises have unique vocabularies such as product names, off-the-shelf spelling solutions fall short of users' needs.

In this work, we build a multilingual spellchecker that is extremely fast and scalable and that adapts its vocabulary and hence speller output based on a specific product's needs. Furthermore our speller out-performs general purpose spellers by a wide margin on in-domain datasets. Our multilingual speller is used in search in Adobe products, powering autocomplete in various applications.

CCS CONCEPTS

• **Applied computing → Document management and text processing**.

KEYWORDS

spellcheck, spell correction, neural networks, query processing

ACM Reference Format:
Sanat Sharma, Josep Valls-Vargas, Tracy Holloway King, Francois Guerin, and Chirag Arora. 2023. Contextual Multilingual Spellchecker for User Queries. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3539618.3591861

1 INTRODUCTION

Spell correction is a widely studied problem in search and NLP research. Spellcheckers generally comprise two parts: creating a list of candidate corrections and ranking those candidates. Most widely used spellcheckers are built for English and utilize behavioral [12] and/or contextual signals [5] for ranking the suggested corrections. Recent works have also utilized other extrinsic data such as search results [4] or public domain multi-word datasets [6] as ranking signals. Although most spellers are built for English, some works have developed custom spellers for non-English languages such as Bengali [8] or Dutch [3]. These works are hard to scale across multiple languages since they are language specific. Most of the work for spell correction has been around correction in sentences or paragraphs where context is plentiful. In such cases, neural models such as transformers and LSTMs perform well since they capture textual context [10]. However, these systems are usually slower than their frequentists counterparts and do not show much improvement in search query cases where textual context is minimal.

Our work takes a best-of-both-worlds approach: We utilize contextual signals such as search results, behavioral data, and phonetic signals to suggest candidates, while incorporating a small neural model for ranking. In addition, we use a suggestion model that is language agnostic and can scale to multiple languages.

We divide the speller into four components: a behavioral data analysis pipeline to finetune the downstream components; a product specific rule engine to correct common errors and provide editorial overrides; a suggester that takes in user queries and suggests potential replacements for incorrectly spelled tokens; and a neural ranker that calculates the probability of the suggested tokens. We evaluate our speller on both general purpose and product specific domains and showcase significant improvement over current methods.

Our approach is currently used in production by the autocomplete feature in Adobe search and is being integrated in Adobe Express and Adobe Stock for online spell correction.

Our main contributions and business impact are:

(1) A novel approach for creating a fast, multilingual spellchecker for search queries
(2) A novel, low latency architecture for deploying and scaling the spellchecker
(3) Significant improvement over widely available state-of-the-art spellcheckers for short user queries

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591861

2 TRAINING DATASETS

Finding public spellcheck datasets is surprisingly hard, with very few benchmarks available for validation. Furthermore, since we require data for training our models, we decided to employ a bootstrapping approach for dataset generation and leveraged crowd workers for manual curation. This section describes how we created the training data, as well as some datasets used for initial internal evaluation. The evaluation datasets are described in section 5.

2.1 Artificially Generated Query Dataset

We extracted user queries from search over Adobe Stock images for English, French and German locales for analysis. Since we use full queries, the model has some context for multi-word queries.

Data Preprocessing: We removed queries with spelling errors from the dataset by applying the updated Hunspell[1] dictionaries to check for spelling errors and then had the remaining queries reviewed by crowd workers. This created our ground truth dataset.

Artificial Injection of Errors: Most spelling errors are due to one of the following reasons: missing a letter, adding a letter, typing an incorrect letter. To create our artificial dataset, for each query in the list of correctly spelled queries, we injected one or more spelling errors using one of the following techniques in a probability-weighted fashion:

(1) Change the order of letters (e.g. "change" to "chnage"; "check" to "chekc"). This the most common spelling error.
(2) Remove or add a vowel (e.g. "malleable" to "mallable" or "malleiable").
(3) Add an additional character (e.g. "fresh" to "freshh" or "frersh").
(4) Replace a character with another character (e.g. "fresh" to "frash" or "frwsh").
(5) Replace accented characters and their unaccented counterparts with another character in the same class (e.g. "français" to "francais"; "wörter" to "worter").
(6) For words with two identical letters in a row, have only one letter (e.g. change "happiness" to "hapiness").

The artificial errors were patterned on real life errors and were weighted at a ratio of 7:5:4:2:7:2 respectively. For addition of vowels, only vowels that usually follow one another were chosen, e.g. for 'e', 'i' was much more likely to be added than 'u'. Each query in the example set had one or more errors injected into them.

Our final artificial dataset size is shown in in Table 1.

English Queries	~1.5M
French Queries	~1.2M
German Queries	~1.4M

Table 1: Training data size

Table 2 shows example input queries and their artificially misspelled training counterparts. Only some words in the query have added errors so that the model also learns to recognize correctly spelled words.

Query	Error Tokenized Query	Error Type
atlantic mackerel	[agtlantic, mackrel]	LETTER_ADD_REMOVE VOWEL_ADD_REMOVE
burgundy background	[burgundy, backgrround]	DOUBLE_ADD_REMOVE
glacier national park and hike	[glaicer, natoinal, 0ark, and, hik]	LETTER_ORDER LETTER_ORDER LETTER_CHANGE LETTER_ADD_REMOVE
medal icon	[,edal, icon]	LETTER_CHANGE

Table 2: Examples of spelling errors introduced to naturally occurring, correctly spelled Adobe Stock queries. Note that punctuation marks and numbers can substitute for letters.

2.2 Birkbeck Corpus

The Birkbeck corpus[2] contains 36,133 misspellings of 6,136 words. It is an amalgamation of errors taken from the native-speaker section (British and American writers) of the Birkbeck spelling error corpus, a collection of spelling errors gathered from various sources, available with detailed documentation from the Oxford Text Archive.[3] It includes the results of spelling tests and errors from free writing, primarily from schoolchildren, university students and adult literacy students. We utilize 18,295 misspellings from Birkbeck as part of our English training dataset.

2.3 Commonly Misspelled Word Corpora

The Aspell [1] corpus contains ~1500 common misspellings. Wikipedia[4] lists commonly misspelled words. We used these to mine for queries in our domain that feature these misspelled words and for internal evaluation for model selection.

3 MODEL

Following common practice, we divide the spellcheck model into two modules: a suggester module and a ranker module. The suggester module takes in the user query and suggests possible correction tokens for any incorrectly spelled tokens. The ranker module ranks the suggestions and outputs the most probable candidate. This is shown in Figure 1.

3.1 Symmetric Delete Suggester

We utilize the Symmetric Delete[5] [7] algorithm for our suggester module. Symmetric Delete generates a permutation index for words in the dictionary at index time. Instead of calculating transposes + replaces + inserts + deletes at runtime, Symmetric Delete only calculates deletes of the index dictionary. The symmetric delete suggester has two key advantages:

- **Latency**: The module is extremely fast for up to 2 edit distances, with an average of ~1ms latency. This is critical for

[1] http://hunspell.github.io/

[2] http://www.dcs.bbk.ac.uk/~ROGER/corpora.html

[3] http://ota.ahds.ac.uk/

[4] https://en.wikipedia.org/wiki/Commonly_misspelled_English_words

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

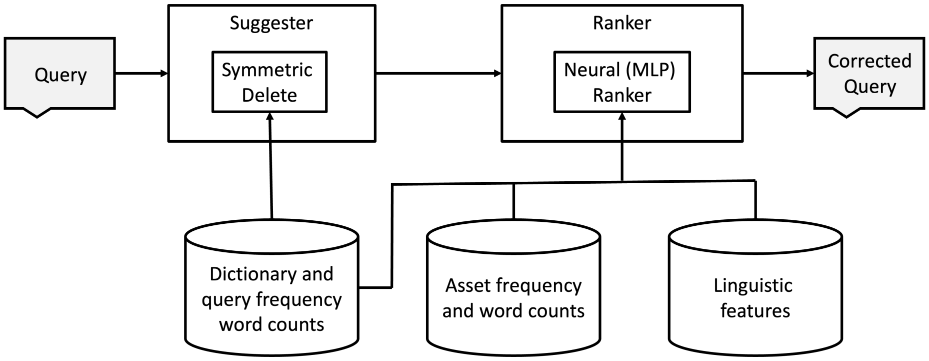

Figure 1: Model architecture of the speller

query spell correction. The speed comes from inexpensive delete-only edit candidate generation and pre-calculation.

- **Language Agnostic**: The module is language agnostic, not requiring language specific characteristics to generate suggestions.

Index Time Operation. At index time, we utilize a dictionary of correct words and generate the symmetric delete index from those. The dictionary of correct words is generated from known language dictionaries, including FastText [2] word dictionaries, Adobe-specific product terms (e.g. product names, file extensions) and behavioral data (e.g. popular queries). The addition of custom vocabulary is important because most enterprises have custom language that is not supported by the open source dictionaries.

Runtime Operation. At runtime, given a user query, we first check if the query is correctly spelled. If it is incorrect, we find all candidates within 1 edit distance. If <3 candidates are generated, we then utilize 2 edit distance suggestions. This balances speed and precision, as increasing the edit distance leads to more suggestions but higher latency. In our analysis of Adobe user queries, we found that 88% of spelling errors are 1 edit distance away. So, 2 edit distance suggestions are used sparingly.

3.2 Neural Ranker

We utilize a neural network to rank the suggestions from the suggester module. Due to our low latency requirements, we use a multilayer perceptron network (MLP) rather than recurrent neural nets or transformers. Our MLP consists of 5 fully connected layers, with dropout and batch normalization. Since MLPs do not do well at token level understanding, we utilize the features for each suggestion rather than the tokens themselves in order to improve performance on unseen words (i.e. unique spelling errors).

The features we utilize for each suggestion are below. All features were scaled and normalized (0–1) before being fed to the neural network.

- **Word Count**: In most cases, we want to recommend more common words. We store the number of occurrences of each word in the query set. The word counts vary based on application, enabling per-application suggestions.
- **Asset Frequency**: In most cases, we want to correct to a word which retrieves more search results. For each word, we store the number of assets associated with it. This feature is application specific.
- **Download Count**: Query success is indicated by downloads in Adobe Stock. We store the number of downloads for the first 100 (first page) results for each word. This feature is only used on Adobe Stock.
- **Levenshtein Distance**: Standard string edit distance measurement [11].
- **Language Locale**: Language of the locale the query is issued on (e.g. French, Japanese).
- **Application**: Which application the query is scoped to (e.g. Adobe Stock, Adobe Express).
- **Phonetic Similarity**: For misspellings where the misspelled word is phonetically correct (e.g. muzeem vs. museum), this feature helps focus on phonetically similar corrections.[6]

4 SERVICE ARCHITECTURE

In order to serve, scale and maintain low latency for the spellchecker, we implemented a novel architecture (Figure 2). The suggester module can struggle with task-specific multi-word errors caused by compounding or decompounding (e.g. "creativecloud" (creative cloud) and "photo shop express" (photoshop express)). We created a multi-word expression (MWE) module that corrects the most custom multi-word errors. This module uses a key-value map based on the most common queries in Adobe products and can be different for different applications. We also created a behavioral pipeline that automatically updates the statistics for the model features (e.g. asset frequency, word count). This updates the speller based on user data without the need for extrinsic changes (e.g. automatically incorporating new words like "covid" and "blockchain").

5 EVALUATION

We performed several qualitative and quantitative evaluations on a variety of datasets. Here, we present results from two different applications to demonstrate the ability of the speller to adapt to different query patterns.

[6] This feature was only utilized for English.