Testing ~~spellcheckers~~ spell checkers

Ekaterina Garanina

1 Short intro

Spelling correction is a required task for many applications (writing assistance, information retrieval, messaging), and there are several well-known tools for spell checking (Aspell, Hunspell, JamSpell, among others). Concerning the data for developing the systems though, the resources are quite scarce, especially for context-based correction. There are available learner corpora (Bryant et al., 2019), clinical data (Johnson et al., 2016; Lu et al., 2019), data from search engines (Hagen et al., 2017), and word pair collections[1] but the most prominent way to obtain vast amount of data is artificial noising with various strategies.

In this work, I will evaluate several available spell checking tools based on news dataset with noising. I try to adhere to the end user point of view and evaluate systems end-to-end with sequence accuracy.

2 Data

As my dataset, I take 10K news sentences from 2020 (Goldhahn et al., 2012) noised with algorithms from (Jayanthi et al., 2020): substitution of words with their common misspellings collected from available sources and character-level error sampling based on the search engine misspellings logs. I remove all examples where noising causes different tokenization (500) for further tokenization issues exploration.

3 Tools

Hunspell[2] is a very widely-used open-source spell checking tool, which was initially developed for languages with rich morphology. Its algorithm is based on dictionaries and extensive morphological information, such as prefixes, suffixes, and inflections.

[1] https://www.dcs.bbk.ac.uk/~roger/corpora.html
[2] http://hunspell.github.io

Jamspell[3] is a context-based spell checker which uses Peter Norvig's[4] algorithm optimized with SymSpell[5] approach for generating candidates and a statistical language model to account for context during selection.

NeuSpell[6] (Jayanthi et al., 2020) is based on neural networks trained on a sequence labelling task, where the model is supposed to output probability distribution over vocabulary for each input token. The released models include BERT and LSTMs on various embeddings trained on a synthetic corpus noised with several patterns.

I have also tried two more approaches:

Contextual Spell Check module from SpaCy[7], which is another approach which uses BERT. It masks OOV tokens and outputs the best candidate from top model predictions using Levenstein distance.

T5 for Grammatical Error Correction[8] is a sequence-to-sequence model for general grammar error correction.

However, I did not include them in my further analysis because, apart from demanding a lot of time and GPU resources, they produced unsatisfactory results in my preliminary experiments.

4 Evaluation

For evaluation, I use sequence accuracy, which measures the percentage of exact matches of predicted sentences with gold standard. I decided to go with sequence accuracy for several reasons:

1. Given that I test spell checkers "out-of-the-box", which means passing a raw string as an input and also varying models' capabilities on splitting and merging words, it becomes non-trivial to map tokens for token-based evaluation.
2. I consider a sequence-based metric the strictest one and also the closest to user's expectations, which is in line with my experimental setup.

5 Experiments

Although aiming at minimal interference into text processing by the tool, I had to do some processing:

[3]https://github.com/bakwc/JamSpell
[4]https://norvig.com/spell-correct.html
[5]https://github.com/wolfgarbe/SymSpell
[6]https://github.com/neuspell/neuspell
[7]https://spacy.io/universe/project/contextualSpellCheck
[8]https://huggingface.co/vennify/t5-base-grammar-correction

Tool	**Seq Acc**
Hunspell	0.104
NeuSpell	0.27
JamSpell	**0.271**

Table 1: Sequence accuracy for three spell checking tools.

1. for Hunspell, I tokenized the text with customized SpaCy tokenizer, applying all rules except for splitting by apostrophes.
2. for Neuspell, I had to detokenize the output since the tool outputs tokenized sequences joined by whitespace.

It should also be noted that the noising strategy I use is used for Neuspell training (though on different data), so I expect a decent accuracy score for this tool. Moreover, JamSpell's training data contains news corpus but from different years and the noising strategy is different.

6 Results

The results are presented in Table 1.

In addition, I randomly selected 50 examples of errors for each tool and observed some patterns.

Hunspell has problems with proper names and noised tokens with large edit distance from the original:

Neuspell. Very good, as expected given the training data, but major failure at detokenization. My efforts to restore it were not very successful, e.g. resulting in leaving hyphenated words split. It can also hallucinate if some punctuation is not in the dictionary.

JamSpell. Also problems with long edit dispances between clean and noised texts. No major problems with names (maybe because of the domain)

7 Discussion

I think that the main challenge of spell checker overview is to actually run existing tools since they are written in different languages, require different setup and input format, and output predictions in a different way. Sometimes it's required to fight with bugs, which makes using the tool in the off-the-shelf manner pretty impossible.

One more major challenge is the scarcity of well-compiled corpora because existing corpora are specialized (learner corpora, queries, etc) and preprocessed in different ways (concerining tokenization, casing, annotation). Noising is also non-trivial because the typos and misspellings are mostly non-random and have certain patterns.

The more technical problem is tokenization and token-based evaluation since both noising and correction can produce different tokenizations compared to gold standard. The alignment becomes even more difficult with recent sequence-to-sequence models which are not tied to input tokens.

In my work, I tried to tackle this challenges by unifying the interface for working with different tools, taking existing noising strategy based on observed misspelling patterns, and applying a strict sequence-based accuracy measure. I believe that context can be extremely helpful for a spell checker but modern context-aware models should somehow be prevented from hallucination which they're prone to.

References

Bryant, C., Felice, M., Andersen, Ø. E., and Briscoe, T. (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.

Goldhahn, D., Eckart, T., and Quasthoff, U. (2012). Building large monolingual dictionaries at the Leipzig corpora collection: From 100 to 200 languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 759–765, Istanbul, Turkey. European Language Resources Association (ELRA).

Hagen, M., Potthast, M., Gohsen, M., Rathgeber, A., and Stein, B. (2017). A large-scale query spelling correction corpus. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1261–1264.

Jayanthi, S. M., Pruthi, D., and Neubig, G. (2020). NeuSpell: A neural spelling correction toolkit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 158–164, Online. Association for Computational Linguistics.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., and Mark, R. G. (2016). Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

Lu, C. J., Aronson, A. R., Shooshan, S. E., and Demner-Fushman, D. (2019). Spell checker for consumer language (CSpell). *Journal of the American Medical Informatics Association*, 26(3):211–218.