Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 697–706
Marseille, 20-25 June 2022
© European Language Resources Association (ELRA), licensed under CC-BY-NC-4.0

LeSpell – A Multi-Lingual Benchmark Corpus of Spelling Errors to Develop Spellchecking Methods for Learner Language

Marie Bexte, Ronja Laarmann-Quante, Andrea Horbach, Torsten Zesch
Research Cluster "Digitalization, Diversity and Lifelong Learning – Consequences for Higher Education",
FernUniversität in Hagen, Germany

Abstract

Spellchecking text written by language learners is especially challenging because errors made by learners differ both quantitatively and qualitatively from errors made by already proficient learners. We introduce LeSpell, a multi-lingual (English, German, Italian, and Czech) evaluation data set of spelling mistakes in context that we compiled from seven underlying learner corpora. Our experiments show that existing spellcheckers do not work well with learner data. Thus, we introduce a highly customizable spell checking component for the DKPro architecture, which improves performance in many settings.

Keywords: spellchecking, error correction, language learner data, multilingual

1. Introduction

Learner texts often differ both in the quantity and quality of errors from texts written by native speakers (Rimrott and Heift, 2008; Flor et al., 2015). Consider the following example, taken from the MERLIN-DE corpus (see Section 2): *Ich schuche 3-4 zimm in Dreter Stock*. The orthographically correct version would be *Ich suche 3-4 Zimmer in dritter Stock.* ('I am looking for 3-4 rooms on third floor.') Note that grammatical errors are not of interest here. We can see that all misspellings in this example have an edit distance greater than 1 to their corrections. The pronunciation, however, is often still very similar (e.g. *Dreter* [dʀeːtɐ] and *dritter* [dʀɪtɐ]). Proficient writers, in contrast, often produce typos which result in misspellings such as *driktter*, where an unrelated incorrect letter is introduced.

Consequently, many downstream NLP tools are known to decrease in performance when applied on non-standard data such as learner language (Foster, 2010; Keiper et al., 2016). Off-the-shelf spellchecking tools are mostly targeted at errors of proficient writers and fail to correct multi-edit errors produced by learners (Rimrott and Heift, 2008). Therefore, there have been several attempts to build spellcheckers that are specialized in learner errors, both for second language (L2, e.g. Boyd (2009)) and first language (L1) learners, i.e. young children (e.g. Stüker et al. (2011), Downs et al. (2020)). The different approaches are hard to compare, however, because they have been evaluated on different data sets and partly on different languages. Therefore, it has not yet been determined to what extent the approaches are transferable to other learner populations and other languages.

In general, Flor et al. (2019) noted that most spellcheckers have been evaluated on either proprietary or artificial data. Therefore, they released TOEFL-Spell, a publicly available benchmark spelling data set which is based on real errors from L2 learners of English. However, there is yet no comparable benchmark set for other languages or even for multiple languages sharing a common format.

Also, general-purpose spellcheckers have mostly been evaluated and compared on English data only (see e.g. Näther (2020) for a comparison of 14 spellcheckers on artificially created English data). To begin with, only few spellcheckers are freely available off-the-shelf for several languages. Popular exceptions are Hunspell[1] and LanguageTool.[2] Some of the other available tools have not been specifically designed for a particular language. They claim to be transferable, but they have not been trained and evaluated on non-English data (e.g. NeuSpell (Jayanthi et al., 2020)) or only paid versions of a spellchecker come with models for different languages (see Jamspell).[3]

It is also an open question how well neural models will work for learner data. Not only is data containing learner misspellings rare, it is also hard to synthesize. Even with a fine-tuned neural model available, a drawback of it would still be that it is less transparent, meaning that the main way of influencing its performance is the choice of training data and that it is not as straightforward to separately evaluate error detection and correction.

Contributions Our goal is to foster comparable, multilingual research on language learner errors as well as easy integration of spellchecking methods for language learners into NLP pipelines. To this end, our paper makes the following contributions:

1. We introduce **LeSpell**, a multilingual data set of spelling errors by language learners, which consists of spelling errors in context taken from texts of L1 and L2 learners of varying proficiency levels from four different languages (English, German, Italian and Czech). The data set is derived from seven existing learner corpora and transferred to

[1] http://hunspell.github.io/
[2] https://github.com/languagetool-org/languagetool
[3] https://jamspell.com

a uniform XML-format. We make all data publicly available.[4] In cases where the license of a corpus does not allow re-distribution, we provide the code to transform the original corpus into the LESPELL format.

2. We release the **highly-customizable spellchecking software** used in our experiments as DKPro Spelling, a Java-based toolkit which can be integrated into any DKPro pipeline as well as used as an independent preprocessing step for any NLP project. Crucially, DKPro Spelling can be used for spelling error detection and correction in any language for which a dictionary, a tokenizer and a Named Entity Recognition (NER) module are available. Our spellchecking toolkit can be used in two ways: In the **user mode**, one can easily apply the default setting via a Java API. In the **expert mode**, the expert user may supply their own resources (such as customized cost matrices for Levenshtein distance or their own language models) and directly integrate the spellcheckers as components into their own UIMA or DKPro Pipelines (Eckart De Castilho and Gurevych, 2014).

3. We evaluate DKPro Spelling against existing multilingual spellchecking tools (Hunspell and LanguageTool) on our LESPELL data set. We discuss parameters that can be varied within a spellchecker and analyze their influence on the spellchecking performance. Our experiments lead to a default configuration of DKPro Spelling which outperforms both Hunspell and LanguageTool in most settings, in particular for languages other than English and learners with non-native proficiency.

2. Multilingual Spelling Error Benchmark Data Set

In order to construct the multilingual LESPELL benchmark data set for spelling error detection and correction, we extract misspelled words and their contexts from a number of learner corpora that have been annotated for spelling errors. Our collection covers four European languages (English, German, Italian, and Czech), different modalities (hand- and typewritten), and both L1 and L2 speakers. We are interested only in orthographical errors and thus only consider corpora that differentiate between orthographical and grammatical errors. In the following, we briefly characterize the corpora that make up the LESPELL data set and describe how we extracted the spelling errors.

2.1. Source Corpora

TOEFL-Spell (Flor et al., 2019) is based on TOEFL11, the ETS Corpus of Non-Native Written English (Blanchard et al., 2013) and was introduced as a benchmark corpus of English misspellings. It consists of spelling errors found in argumentative essays written by advanced L2 learners of English with different language backgrounds. The essays were written for the TOEFL® iBT test, which measures academic English proficiency.

MERLIN-DE, MERLIN-IT, MERLIN-CZ (Wisniewski et al., 2018) consist of texts that were written by L2 learners of German, Italian and Czech, respectively. The texts were part of TELC[5] language tests (for German and Italian) and exams of the University of Prague (for Czech) at the levels A1 to C1 of the Common European Reference Frame. We extracted all errors that were annotated as orthography errors, covering spelling, capitalization and word-building (i.e. missing and superfluous whitespaces between words) problems.

Litkey (Laarmann-Quante et al., 2019) consists of texts written by German primary school children between grades 2 and 4. The texts were produced freely based on short picture stories. The corpus contains a manually constructed target hypothesis which corrects orthographic errors only.

SkaLa (Scholten-Akoun et al., 2014) results from a test designed to assess first-year university students' academic writing skills. Specifically, we use a set of over 2000 essays written by students in a study program for teachers in Germany, i.e. many of the test-takers were native speakers. As this data set does not come with error annotation, we collected 600 instances of words not in a dictionary and manually annotated them as spelling errors or not, including a manual correction. This means that no real-word errors are marked in this data set.

CItA (Barbagli et al., 2016) contains texts produced by Italian children during the first two years of lower secondary school. The texts are based on different writing prompts (e.g. narrative, argumentative, descriptive). We normalize the set of different characters used in the CItA corpus with respect to apostrophes and quotation marks to ensure proper tokenization. We extract as spelling errors all errors that have been manually annotated as orthographic, except for erroneous usages of apostrophes.

2.2. Data Set Overview

Table 1 provides an overview of the individual subsets in our benchmark spelling data set. Under *modality*, we distinguish whether the original data was collected hand- or computer-written. The average Levenshtein distance refers to the distance between an erroneous word and its gold correction. Errors in which a word was erroneously split into two or more parts by the writer are excluded from this statistic. The column *split words* shows the proportion of errors that include such an erroneous splitting of words.

[4] https://github.com/ltl-ude/ltl-spelling

[5] https://www.telc.net

Data Set Name	Language		Modality	# Errors	ø Levenshtein	Error Rate	Split Words
TOEFL-Spell	English	L2	typed	6,251	1.25	2.19%	0.5%
MERLIN-DE	German	L2	hand-written	5,366	1.34	6.94%	12.9%
Litkey	German	L1	hand-written	38,698	1.40	20.20%	11.0%
SkaLa	German	L1	typed	423	1.25	3.40%	0.9%
MERLIN-IT	Italian	L2	hand-written	2,137	1.18	3.22%	3.0%
CItA	Italian	L1	hand-written	1,431	1.14	0.39%	4.1%
MERLIN-CZ	Czech	L2	hand-written	4,807	1.35	9.95%	3.1%

Table 1: Source corpus information and statistics

We convert all corpora into a uniform XML-format. Details about the format can be found in the documentation that comes with the data set. Note that for all corpora, correct sentences are included as well, so that our data set can be easily used for developing and testing spelling error correction methods that take the whole text into account. We directly provide the benchmark data for MERLIN (all languages) and Litkey. For the other corpora, due to licensing issues, we only provide the code for transforming the original format into the XML format of the spelling error benchmark data set.

3. Experimental Setup

We now discuss individual preprocessing toolchains per language, describe Hunspell and LanguageTool, which we used as baseline systems, and then present our own spellchecking toolkit DKPro Spelling.

3.1. Preprocessing

We use language-specific preprocessing tools provided by DKPro where possible. For Czech, there is no segmenter and NER model available, so we use the English OpenNlp segmenter and the English NER model instead. While there is no NER model available for Italian either, there is an Italian POS model and we use this to identify Named Entities (NEs) via the respective POS tag. For the Litkey corpus, we use a whitespace segmenter because it already comes in a tokenized format.

3.2. Existing Multi-lingual Spellcheckers

Two popular spelling correction tools, which are freely available for multiple languages, are Hunspell and LanguageTool. In the following experiments, we apply Hunspell and LanguageTool to our multilingual data set of language learner corpora. We then experiment with various parameters that might influence spellchecking performance.

The spelling correction process can be seen as divided into two steps. The first one is to identify which words of a corpus are considered to be misspellings by the respective tool. In a second step, correction candidates are then selected for the misspellings in a text.

To gain insight into how these two aspects affect the performance of the spellchecking tools, we consider them separately.

Hunspell can be used for any language for which a dictionary is available. The dictionaries can contain additional information about replacements of common errors, keyboard layout, or pronunciation, hence the tool's spellchecking ability depends a lot on the quality of the dictionary.[6] The tool uses a dictionary lookup for spelling error detection and different means for generating correction candidates: applying manipulations to the misspelled word itself and computing shared n-grams between the misspelling and dictionary words. Phonetic information is rarely present in the commonly available dictionaries and therefore mostly not used. A general limitation of Hunspell is that it only looks at individual words and does take neither frequency of correction candidates nor context into account.

LanguageTool currently supports language checking including orthography, grammar and style for over 30 languages and language variants. It mainly relies on hand-crafted rules which can take the local context into account. Note that LanguageTool does not support Czech yet, meaning that we cannot report its spellchecking performance on MERLIN-CZ.

3.3. DKPro Spelling

We design our spellchecking tool as part of the DKPro framework family. Its main feature is the customizability according to the characteristics of the data that shall be corrected. All language resources are configurable, which makes the tool language-agnostic. Its functionality comprises non-word spelling error detection and correction.

Our architecture follows earlier approaches like (Flor and Futagi, 2012) in combining different candidate generation methods and re-ranking the candidates by looking at the context. However, our system is freely available and our experiments evaluate it on non-artificial multi-lingual data. Figure 1 illustrates the pipeline, which we describe in the following.

Misspelling Detection The texts are first tokenized and annotated for NEs, numerals and punctuation, all of which are not considered as potential misspellings in the default setting.

[6] see Spylls (https://github.com/zverok/spylls) and the corresponding series of articles (https://zverok.github.io/spellchecker.html) for further information on Hunspell

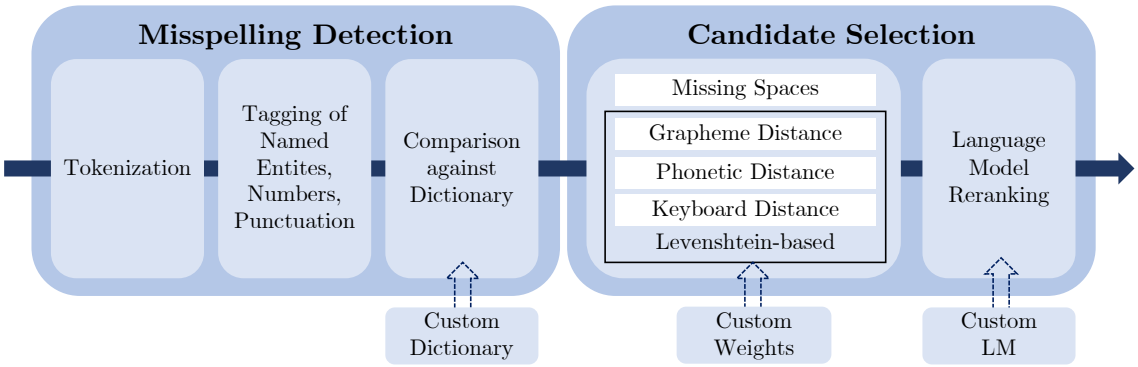

Figure 1: Overview of the DKPro Spelling pipeline.

Our toolkit uses existing tokenizers and NER systems provided by DKPro Core (Eckart de Castilho and Gurevych, 2014). For the identification of numerals and punctuation, it relies on regular expression matching. A subsequent step compares the remaining words against a full-form dictionary (or a set of dictionaries) and marks all words which are not present in any of them as misspellings. A number of heuristics is used to allow capitalization of tokens at the beginning of sentences, after line breaks, or after quotation marks.
We provide default dictionaries for English, German, Italian, and Czech. A user can easily plug in their own additional word lists for task-specific vocabulary or replace the dictionary altogether.

Candidate Selection Our toolkit integrates different candidate generation approaches as separate modules, which can be combined if needed. Each method ranks the generated candidates and returns the best n, where n can be set by the user. In case of ties, more than n candidates may be returned to include all candidates with equal rank. When inserting spaces, if any, there are typically only a few candidates found, hence less than the specified n candidates may be returned.
All Levenshtein-based methods (grapheme-, phoneme- and keyboard-based) can be used with custom weight or distance matrices. We provide English, German, Italian, and Czech keyboard distances which contain the number of keys between each pair of letters, with adjacent keys having a distance of 1.

Candidate Re-ranking To re-rank the generated candidates, the user can plug in any Web1T language model or a list of unigram frequencies. The default n-gram size to consider for this step is 3, but can be changed by the user. If no re-ranking is desired, this step can simply be skipped.

4. Misspelling Detection

Table 2 shows results for error detection using several dictionary setups on the sub-corpora in our data set. We report precision, recall and F-score for the class 'error'. Note that the SkaLa corpus is not included here because we do not have gold-standard error annotations, as explained in Section 2.1.

Lang	Corpus	Dict.	Size	P	R	F
EN	TOEFL	Lang-tool	n/a	.72	.94	.82
	TOEFL	Hun-tool	n/a	.81	.99	.89
	TOEFL	Hun-dict	123k	.52	.98	.68
DE	MERLIN	Lang-tool	n/a	.69	.59	.64
	MERLIN	Hun-tool	n/a	.64	.62	.63
	MERLIN	childLex	45k	.37	.64	.47
	MERLIN	Hun-dict	727k	.44	.73	.55
	Litkey	Lang-tool	n/a	.70	.66	.68
	Litkey	Hun-tool	n/a	.74	.72	.73
	Litkey	childLex	45k	.77	.74	.76
	Litkey	Hun-dict	727k	.84	.75	.79
IT	MERLIN	Lang-tool	n/a	.55	.64	.59
	MERLIN	Hun-tool	n/a	.53	.69	.60
	MERLIN	Hun-dict	396k	.38	.56	.46
	CItA	Lang-tool	n/a	.12	.44	.18
	CItA	Hun-tool	n/a	.11	.47	.18
	CItA	Hun-dict	396k	.15	.45	.23
CZ	MERLIN	Hun-tool	n/a	.67	.82	.74
	MERLIN	Hun-dict	313k	.66	.82	.73

Table 2: Error Detection Results

In the column *dictionary*, **Hun-tool** denotes the application of Hunspell on a word to check whether it is deemed correct or not.[7]
Similarly, **Lang-tool** indicates the use of LanguageTool to determine whether or not a word is marked as a misspelling, which we operationalize as one of its rules of type *misspelling* firing.
Hunspell dictionaries are designed with proficient writers in mind. However, both children and language learners have a restricted vocabulary. Hence, very rare or specialized words, which are contained in Hunspell, may rather be real-word errors, i.e. misspellings of more frequent words. For example, *Spas* in a child's text might likely be a misspelling of *Spaß* 'fun' rather than the plural of the word *Spa* 'spa'. Therefore we run our tool with a dictionary more suited to L1 German language learners. The **childLex** (Schroeder et

[7]For our experiments, we use the Hunspell dictionaries provided at https://github.com/wooorm/dictionaries.

al., 2015) dictionary comes as a full-form word list and consists of 45k types found in 500 children's books. We hypothesize that using this dictionary to detect errors yields a higher recall and comparable precision for Litkey and potentially also MERLIN-DE. Note that there are differences in the acquisition order of vocabulary between L1 and L2 learners (Brysbaert et al., 2021), therefore our intuitions about the effect of childLex on MERLIN-DE are less clear.

A Hunspell dictionary comes as a combination of a dictionary (.dic) and an affix (.aff) file, which allows to create inflected and derived forms of a word (e.g. *show, shows, showing, showed*). Because the childLex dictionary is a word list, we create full-form dictionaries out of the .dic- and .aff-files to use these in our tool as another baseline to compare the performance on the full-form childLex dictionary to. To estimate how much improvement using Hunspell itself gives over using the full-form Hunspell wordlist, we do this for all languages. We create these full-form Hunspell dictionaries using Hunspell's *wordforms* command. The resulting dictionaries are described as **Hun-dict** in Table 2. Note that these dictionaries do not include all compounds that can be built from the words in the original Hunspell dictionary file.

Discussion We observe that *Hunspell-tool* and *LanguageTool* perform similarly on all corpora, the only exceptions being TOEFL-Spell and the Litkey corpus, for which *Hunspell-tool* achieves higher scores. Both *LanguageTool* and *Hunspell-tool* outperform *Hunspell-dict* on TOEFL-Spell, Merlin-DE, and MERLIN-IT. On MERLIN-CZ, using *Hunspell-tool* over the full-form Hunspell dictionary only gives a minor increase of .01 in precision.

Using the more learner-suited *childLex* instead of *Hunspell-tool* or *LanguageTool* yields much lower precision on MERLIN-DE, but increases performance on the Litkey corpus. What increases performance on the Litkey corpus further is to use *Hunspell-dict*, as it leads to an increase in precision. The childLex dictionary therefore seems too small, thus erroneously flagging tokens not contained in it as errors, which are however contained in *Hunspell-dict*. The reason for both our approaches (*childLex* and *Hunspell-dict*) outperforming *LanguageTool* and *Hunspell-dict* on the Litkey corpus is due to the fact that we do not consider Named Entities as possible misspellings. A protagonist in the Litkey texts is named 'Dodo', which leads to around 7000 false positives for both *LanguageTool* and *Hunspell-tool*. Filtering Named Entities can also explain the slightly better performance of our *Hunspell-dict* on the CItA corpus.

Overall, we see a large variance in error detection performance between the different data sets. The highest F-Score achieved is .89 for the English TOEFL-Spell corpus and the lowest F-Score is .23 for the Italian CItA corpus. There is also a large variance between corpora of the same language.

As one would expect, among the false positives for *Hunspell-tool*, i.e. correct words erroneously flagged as errors, we find undetected NEs (e.g. *Ghandi* [CItA], *Dodo* [Litkey], *Frauenkirche* [MERLIN-CZ]), segmentation errors (e.g. *28.maggio* [MERLIN-IT], *4years* [TOEFL-Spell]) as well as foreign words, colloquial terms, abbreviations or interjections not covered by the respective dictionary (e.g. *baby-sitter* [MERLIN-IT], *nix* [Litkey], *CO2* [TOEFL-Spell], *hahaha* [CItA]). For German, there are still compounds that were not recognized as such (e.g. *Spielzeugkiste* "toy box" [Litkey], *Au-pair-Agentur* "au pair agency" [MERLIN-DE]). Some words are flagged as misspellings by the system though they are not annotated as orthographic errors in the corpora but rather as grammatical errors. These are counted as *false positives* as well. This accounts for 11% of the false positives in CItA and 32% in Litkey.

There are particularly many false positives for the Italian CItA corpus. Partly, this is due to annotation problems in the original corpus, i.e. misspellings that were not marked by the original annotators (accounting for about one third of 100 randomly selected instances marked as false positives). In addition, there are several English words, proper names and colloquial terms in the texts that were also not covered by the dictionary.

The false negatives, i.e. erroneous words nevertheless occurring in the dictionary, constitute real-word errors. In the German corpora, many real-word errors are caused by learners splitting up a compound (e.g. *Praktikum Stelle* for *Praktikumsstelle*). For Italian and Czech, we often find missing or wrong accents leading to another existing grammatical form of a word (e.g. *penso* vs. *pensò* [MERLIN-IT], *libí* vs. *líbí* [MERLIN-CZ]). For the English TOEFL-Spell corpus, several false negatives are errors within NEs, which our tool by default never considers as spelling errors. In some cases, the misspelling of a capitalized sentence-initial non-NE word confused the NE recognition and led to similarly unrecognized errors.

Our overall findings are as follows: Error detection for languages other than English works considerably less reliably across data sets and dictionaries. When using a dedicated dictionary geared towards children, performance is higher than that of *Hunspell-tool* on the Litkey corpus, which contains children's texts, but beat by using a full-form dictionary and NE-filtering. For L2 learner language, using *childLex* leads to a decrease in performance compared to all other dictionaries. While the performance drop itself was unexpected, the order between the two data sets confirms that words known by L1 and L2 language learners are substantially different.

5. Error Correction

For error correction, we assume gold standard error detection, i.e. we use every spelling error that is annotated in a corpus instead of those found by the error detection component.

Corpus	Hunspell w/o reranking	Hunspell w reranking	LangTool w/o reranking	LangTool w reranking	DKPro w/o reranking	DKPro w reranking	DKPro Upper Bound
TOEFL	.68	.75	**.75**	**.79**	.60	.70	.94
MERLIN-DE	.36	.47	.46	.46	**.55**	**.59**	.78
Litkey	.31	.37	**.36**	.24	.25	**.42**	.80
SkaLa	.69	**.56**	**.73**	.54	.37	.40	.79
MERLIN-IT	**.49**	.59	.38	.54	.41	**.72**	.83
CItA	**.35**	.47	.27	.37	.30	**.50**	.78
MERLIN-CZ	**.23**	.54	-	-	.22	**.61**	.86

Table 3: Error correction results. We report recall@1 based on gold detection results and compare performance without and with re-ranking candidates using Web1T. Best results per data set with and without reranking in bold. Note that we use a probabilistic recall@1 when re-ranking.

5.1. Baseline Comparison

First, we compare Hunspell and LanguageTool against a simple edit-distance baseline. For that, we use the full-form hunspell dictionaries to select correction candidates with a minimum Damerau-Levenshtein distance to a misspelling. The upper bound for the correction performance is determined by the dictionary. Corrections which are not in the dictionary cannot be found by the spellchecking component. This applies, for example, to some proper names (*Dhirubhai*), compound words (*self-satisfaction*) or technical vocabulary (*geomorphology*). Table 3 shows correction results measured in recall@1, meaning that any misspelling for which the respective method returned the desired correction as the top candidate was counted towards this metric.

Discussion While Hunspell outperforms LanguageTool on the Italian corpora, LanguageTool performs better on English and German data. Our toolkit is mostly outperformed by both Hunspell and LanguageTool, with the exception of the German MERLIN corpus, where it performs better than both Hunspell and LanguageTool. We see, however, that the upper bound for the correction performance of our toolkit is way above the actual performance of any tool, which means that there is still a lot of room for improvement.

In this experiment, Hunspell and our toolkit consider the misspellings isolated from their context. The context can however influence the likelihood of a certain correction candidate being the desired one. Take for example *she loves riding her horase*, where the misspelled *horase* requires one Damerau-Levenshtein edit to be changed to *hoarse* or *horse*, with only the latter fitting into the context.

5.2. Re-ranking Candidates

To take context information into account, we calculate the language model probability of each suggested correction. We use Web1T (Brants, 2006) trigrams, as Web1T is available for all four languages in our data set. If all candidates have the same probability under the trigram language model (which mostly happens if one of the context words was not seen in the training data of the language model), a back-off strategy to a lower order n-gram model is used.

Results are reported in Table 3. This time, we also take into account when multiple candidates end up in the same rank. Because of such ties, we compute recall@1 in a probabilistic way. For example, if there are four candidates at the top rank, one of them being the desired one, the value for this particular misspelling is not 1 nor 0 but 0.25.

Discussion Generally, re-ranking the candidates is beneficial for most corpora, especially Merlin-CZ, where the correction performance without re-ranking is the worst among all corpora (.30) while the performance with trigram-model-reranking is the third best (.60). The best performance could be reached for MERLIN-IT (.72) and TOEFL-Spell (.70). By contrast, the scores for Litkey remain particularly weak.

The results also show that language model re-ranking is in fact not always beneficial for all tools and all corpora. For example, the texts in the SkaLa corpus (proficient native speakers of German) are best corrected by LanguageTool without language-model re-ranking. This may have to do with the large number of erroneously concatenated words in the corpus.

With the added language model reranking, Hunspell works better than LanguageTool for all corpora but the English one. Our toolkit outperforms Hunspell and LanguageTool on all corpora except for TOEFL-Spell and SkaLa. Both consist of texts written by advanced learners or native speakers. This suggests that compared to the established tools, our toolkit is better at handling errors from learners with lower proficiency and corpora with more varying proficiency levels such as MERLIN.

Similar to our motivation to include childLex in error detection in Section 4, it might be beneficial to base the reranking on a language resource that is more resemblant of learner language. To this end, we follow up with an experiment that uses such resources instead of the Web1T frequencies.

5.3. Re-ranking with Learner-focused Resources

Since it is possible that language learners produce many ungrammatical or uncanonical sentences which are better covered by language models built from spoken language, we carry out another experiment where we use Subtlex to rerank correction candidates. Subtlex contains unigram frequencies created from movie subtitles and is available for English (Brysbaert and New, 2009), German (Brysbaert et al., 2011), and Italian (Crepaldi et al., 2015). For German, we additionally investigate re-ranking with unigram frequencies based

	3 candidates per method				10 candidates per method			
	Web1T 3-gram	Web1T 1-gram	childLex 1-gram	subtlex 1-gram	Web1T 3-gram	Web1T 1-gram	childLex 1-gram	subtlex 1-gram
TOEFL-Spell	.70	.48	-	.47	.58	.27	-	.26
MERLIN-DE	.59	.41	.38	.41	.54	.28	.24	.28
Litkey	.42	.29	.34	.32	.34	.18	.23	.21
SkaLa	.40	.28	.27	.26	.34	.17	.15	.15
MERLIN-IT	.72	.50	-	.58	.66	.37	-	.46
CItA	.50	.27	-	.31	.44	.16	-	.19
MERLIN-CZ	.61	.42	-	-	.55	.29	-	-

Table 4: Error correction results with learner-focused re-ranking. We compare generating three vs. ten candidates to re-rank. Reported values are recall@1.

on childLex. Especially for the Litkey corpus, the word frequencies found in children's books may work even better than Subtlex.

In comparison to a trigram model, working with unigrams has the drawback of missing out on the context. In order to have a fairer comparison with Web1T, we therefore also perform re-ranking with Web1T unigrams.

In addition to using different resources to re-rank, we also explore how the results are affected by using either the top 3 or top 10 correction candidates.

As our approach outperformed both Hunspell and LanguageTool on the majority of the corpora in the previous experiment, we only perform this one within our own toolkit.

Results are displayed in Table 4. To account for ties we again report probabilistic recall@1, just like in the previous experiment.

Discussion We find that across all corpora and all language models, using 3 correction candidates for re-ranking outperforms 10 candidates. This indicates that when too many candidates are generated, the language model may easily favor a wrong one, overriding differences in edit-distance.

Furthermore, the Web1T trigram model yields the best result for all corpora. This means that the context provided by a trigram model outweighs the benefits of unigram frequencies that are more learner-specific.

Among the different unigram models, Subtlex outperforms Web1T in most cases, indicating that frequencies from spoken language are indeed often a better choice for language learners than frequencies from the web, if only unigram frequencies are available. For Litkey, the childLex model outperforms the other unigram models, which indicates that children's word frequencies are best covered by children's books. However, one also has to note that for some corpora (MERLIN-DE, CItA, TOEFL-Spell), re-ranking with unigram frequencies is not beneficial at all.

5.4. Other Methods to Generate Correction Candidates

All experiments so far focused on using our toolkit to generate correction candidates based on their Damerau-Levenshtein distance to the respective misspelling on the grapheme level. For learners, it could be beneficial to consider phonetics (see e.g. Boyd (2009), Stüker et al. (2011), Downs et al. (2020)). Furthermore, for typed texts, the keyboard layout could be a valuable source of information (see e.g. Ahmad and Kondrak (2005)). Words may also be falsely concatenated. In our next experiment, we therefore analyze the following alternatives for the generation of correction candidates:

Phoneme-Based Levenshtein calculates the distances based on phonetic transcriptions, which are obtained from the web service *G2P* of the Bavarian Archive of Speech Signals (BAS) (Reichel, 2012; Reichel and Kisler, 2014).[8] Our hypothesis regarding this method is that it works best for texts written by young children (Litkey and CItA), who often produce phonetically motivated misspellings. The idea behind this is that a misspelling that is pronounced the same or similar to the correct spelling of the intended word but looks very different to it on the surface level (e.g. *peases* for *pieces*) would have a rather large Levenshtein distance to its correct spelling on the character level so that the intended word may not be among the correction candidates. Transcribing both into their phonetic representation would however lead to two strings with a smaller Levenshtein distance, which would thus be a benefit for finding the correct spelling of the misspelled word.

Keyboard Distance makes adjustments to the costs of the edit operations based on the (physical) distance that the affected letters have on a keyboard.

Our hypothesis regarding this method is that it works best for typed texts (TOEFL-Spell and SkaLa) because it captures typos.

[8] https://clarin.phonetik.uni-muenchen.de/BASWebServices/interface/Grapheme2Phoneme

Corpus	Grapheme	Phoneme	Keyboard	Spaces	All	Oracle	Upper Bound
TOEFL	**.70**	.55	.51	.02	.63	.81	.94
MERLIN-DE	**.59**	.54	.45	.00	.56	.68	.78
Litkey	**.42**	.37	.25	.01	.36	.52	.80
SkaLa	**.40**	.33	.31	.28	.39	.71	.79
MERLIN-IT	**.72**	.65	.46	.01	.66	.80	.83
CItA	**.50**	.49	.30	.03	.46	.60	.78
MERLIN-CZ	**.61**	**.61**	.34	.00	.59	.70	.86

Table 5: Comparison of our four different methods of correcting misspellings. Oracle counts a misspelling as corrected whenever any of the methods yields the desired correction. Reported values are recall@1.

Missing Spaces This module looks for missing spaces within tokens to find corrections in cases where whitespace between tokens was omitted, e.g. *thereis* for *there is*.

All In this condition, we combine the candidates from all four generation methods prior to re-ranking.

In this experiment, we compare the performance of the newly considered methods to that of the grapheme-based Levenshtein method used in previous experiments. All experiments use re-ranking with Web1T trigrams, as this proved to be the best-performing approach. In Table 5, we report probabilistic recall@1 results for generating three candidates per method. Note that there are often more than three candidates for evaluation because in case of ties, all generated candidates are kept and then re-ranked.

Discussion We can observe that the grapheme-based Levenshtein module always outperforms the phoneme-based and keyboard-distance-based Levenshtein modules, which falsifies our assumptions that phonetic correction would be better for texts written by children and that keyboard-based correction would be superior for typed texts. For most corpora, grapheme-based Levenshtein alone even yields better results than the *all* condition, where candidates from all methods are combined prior to re-ranking. This indicates that the other correction methods primarily introduce incorrect candidates. Recall that the phoneme-based module relies on an automatic phonetic transcription, which sometimes produces false results, which can cause counterintuitive correction candidates. Using keyboard distance always performs worst, which is unexpected for typed texts (SkaLa and TOEFL-Spell). For these texts, the difference in performance between keyboard distance and the other methods is not as large as for the hand-written corpora, but still, there is no benefit in using this method.

5.5. Oracle Condition

Although none of the methods investigated in the previous experiment outperformed the grapheme-based baseline, manual inspection still found local improvements from each method. We therefore perform an analysis in an **oracle condition**, also displayed in Table 5, where we assume that the right correction is found if it is top-ranked by *any* of the correction methods after candidate re-ranking. With this oracle condition we want to know how well words could be corrected if we knew in advance which correction method was the most suitable one.

The oracle condition yields the best results across all corpora, which means that if we knew which correction method was best for a particular word, we could improve correction performance substantially. Note that oracle performance still has a widely varying performance gap to the upper bound across the different corpora. While for MERLIN-IT, oracle performance almost reaches the upper bound, there is a large gap between the two for the Litkey corpus.

Since we as of yet do not have the ability to do know which method gives the correct candidate for a misspelling of interest, we find that for all corpora, the grapheme-based Levenshtein distance with 3 correction candidates and the Web1T trigram model for re-ranking worked best. We therefore declare this as the tool's default setting.

6. Summary & Future Work

Detecting and correcting spelling errors made by learners remains challenging. Our newly compiled evaluation data set provides a multi-lingual benchmark for fostering the development of customized spellcheckers. Our highly customizable spellchecking extension for the DKPro architecture makes first steps in that direction, as it makes it easy to include other languages, dictionaries or language models and thus enables evaluating the effect different adaptions have on spellchecking performance.

One takeaway of our oracle experiment is that there is room for improvement if a system knew which method for candidate generation to apply for a particular incorrect word. Future work could therefore investigate whether there are any hints in a misspelled word that would allow us to choose the best correction method a priori. We also see mixed results of re-ranking, but our evaluation dataset allows for future experiments with better language models. It also allows to test whether neural spellcheckers have any advantages in a multilingual setting with learner data where training data is usually limited.

7. Acknowledgments

This work was conducted in the framework of the Research Cluster D^2L^2 "Digitalization, Diversity and Lifelong Learning – Consequences for Higher Ed-

ucation" of the FernUniversität in Hagen, Germany (https://e.feu.de/english-d2l2).

8. Bibliographical References

Ahmad, F. and Kondrak, G. (2005). Learning a spelling error model from search query logs. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 955–962, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Blanchard, D., Tetreault, J. R., Higgins, D., Cahill, A., and Chodorow, M. (2013). Toefl11: A corpus of non-native english. *ETS Research Report Series*, 2013:15.

Boyd, A. (2009). Pronunciation modeling in spelling correction for writers of English as a Foreign Language. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, pages 31–36, Boulder, Colorado, June. Association for Computational Linguistics.

Brysbaert, M. and New, B. (2009). Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english. *Behavior research methods*, 41(4):977–990.

Brysbaert, M., Buchmeier, M., Conrad, M., Jacobs, A. M., Bölte, J., and Böhl, A. (2011). The word frequency effect. *Experimental psychology*.

Brysbaert, M., Keuleers, E., and Mandera, P. (2021). Which words do english non-native speakers know? new supernational levels based on yes/no decision. *Second Language Research*, 37(2):207–231.

Crepaldi, D., Amenta, S., Pawel, M., Keuleers, E., and Brysbaert, M. (2015). Subtlex-it. subtitle-based word frequency estimates for italian. In *Proceedings of the Annual Meeting of the Italian Association For Experimental Psychology*, pages 10–12.

Downs, B., Anuyah, O., Shukla, A., Fails, J. A., Pera, S., Wright, K., and Kennington, C. (2020). KidSpell: A child-oriented, rule-based, phonetic spellchecker. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6937–6946, Marseille, France, May. European Language Resources Association.

Eckart de Castilho, R. and Gurevych, I. (2014). A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Flor, M. and Futagi, Y. (2012). On using context for automatic correction of non-word misspellings in student essays. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 105–115, Montréal, Canada, June. Association for Computational Linguistics.

Flor, M., Futagi, Y., Lopez, M., and Mulholland, M. (2015). Patterns of misspellings in l2 and l1 english: A view from the ets spelling corpus. *Bergen Language and Linguistics Studies*, 6.

Flor, M., Fried, M., and Rozovskaya, A. (2019). A benchmark corpus of English misspellings and a minimally-supervised model for spelling correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 76–86, Florence, Italy, August. Association for Computational Linguistics.

Foster, J. (2010). "cba to check the spelling": investigating parser performance on discussion forum posts. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 381–384.

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, October. Association for Computational Linguistics.

Keiper, L., Horbach, A., and Thater, S. (2016). Improving POS tagging of German learner language in a reading comprehension scenario. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 198–205.

Näther, M. (2020). An in-depth comparison of 14 spelling correction tools on a common benchmark. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1849–1857, Marseille, France, May. European Language Resources Association.

Reichel, U. D. and Kisler, T. (2014). Language-independent grapheme-phoneme conversion and word stress assignment as a web service. In R. Hoffmann, editor, *Elektronische Sprachverarbeitung: Studientexte zur Sprachkommunikation 71*, pages 42–49. TUDpress.

Reichel, U. D. (2012). PermA and Balloon: Tools for string alignment and text processing. In *INTERSPEECH*, Portland, Oregon.

Rimrott, A. and Heift, T. (2008). Evaluating automatic detection of misspellings in german. *Language Learning & Technology*, 12(3):73–92.

Scholten-Akoun, D., Mashkovskaya, A., and Tischmeyer, D. (2014). Language competencies of future teachers–design and results of an empirical study. *Applied Linguistics Review*, 5(2):401–423.

Schroeder, S., Würzner, K.-M., Heister, J., Geyken, A., and Kliegl, R. (2015). childLex: A lexical database

of German read by children. *Behavior Research Methods*, 47(4):1085–1094.

Stüker, S., Fay, J., and Berkling, K. (2011). Towards context-dependent phonetic spelling error correction in children's freely composed text for diagnostic and pedagogical purposes. In *Twelfth annual conference of the international speech communication association*.

9. Language Resource References

Barbagli, Alessia and Lucisano, Pietro and Dell'Orletta, Felice and Montemagni, Simonetta and Venturi, Giulia. (2016). *CItA: an L1 Italian Learners Corpus to Study the Development of Writing Competence*. European Language Resources Association (ELRA).

Brants, Thorsten. (2006). *Web 1T 5-gram Version 1*.

Eckart De Castilho, Richard and Gurevych, Iryna. (2014). *A broad-coverage collection of portable NLP components for building shareable analysis pipelines*.

Laarmann-Quante, Ronja and Ortmann, Katrin and Ehlert, Anna and Masloch, Simon and Scholz, Doreen and Belke, Eva and Dipper, Stefanie. (2019). *The Litkey Corpus: A richly annotated longitudinal corpus of German texts written by primary school children*.

Wisniewski, Katrin and Abel, Andrea and Vodičková, Kateřina and Plassmann, Sybille and Meurers, Detmar and Woldt, Claudia and Schöne, Karin and Blaschitz, Verena and Lyding, Verena and Nicolas, Lionel and Vettori, Chiara and Pečený, Pavel and Hana, Jirka and Čurdová, Veronika and Štindlová, Barbora and Klein, Gudrun and Lauppe, Louise and Boyd, Adriane and Bykh, Serhiy and Krivanek, Julia. (2018). *MERLIN Written Learner Corpus for Czech, German, Italian 1.1*.