PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 105 APRIL 2016 77–99

A Comparison of Four Character-Level String-to-String Translation Models for (OCR) Spelling Error Correction

Steffen Eger[a], Tim vor der Brück[b], Alexander Mehler[c]

[a] Ubiquitous Knowledge Processing Lab, Technische Universtität Darmstadt
[b] CC Distributed Secure Software Systems, Lucerne University of Applied Sciences and Arts
[c] Text Technology Lab, Goethe University Frankfurt am Main

Abstract

We consider the isolated spelling error correction problem as a specific subproblem of the more general string-to-string translation problem. In this context, we investigate four general string-to-string transformation models that have been suggested in recent years and apply them within the spelling error correction paradigm. In particular, we investigate how a simple 'k-best decoding plus dictionary lookup' strategy performs in this context and find that such an approach can significantly outdo baselines such as edit distance, weighted edit distance, and the noisy channel Brill and Moore model to spelling error correction. We also consider elementary combination techniques for our models such as language model weighted majority voting and center string combination. Finally, we consider real-world OCR post-correction for a dataset sampled from medieval Latin texts.

1. Introduction

Spelling error correction is a classical and important natural language processing (NLP) task, which, due to the large amount of unedited text available online, such as in tweets, blogs, and emails, has become even more relevant in recent times. Moreover, spelling error correction, in a broader meaning of the term, has also been of interest in the digital humanities where, for instance, large amounts of OCR (Optical character recognition) scanned text of historical or contemporary documents must be post-processed, or, even more generally, normalized (Mitankin et al., 2014; Springmann et al., 2014). In the same digital humanities context, spelling error correction

Corresponding author:
eger@ukp.informatik.tu-darmstadt.de
Cite as: Steffen Eger, Tim vor der Brück, Alexander Mehler. A Comparison of Four Character-Level String-to-String Translation Models for (OCR) Spelling Error Correction. The Prague Bulletin of Mathematical Linguistics No. 105, 2016, pp. 77–99. doi: 10.1515/pralin-2016-0004.

may be important in correcting errors committed by scribes in reproducing historical documents (Reynolds and Wilson, 1991). Beyond error correction, one faces wide ranges of co-existing spelling variants especially in documents of historical languages (e.g., medieval Latin) that must be normalized/standardized in order to be finally mapped to their corresponding lemmas.

Approaches to spelling correction (or standardization) are typically distinguished as to whether they target *isolated word-error correction* or *context-sensitive spelling correction* — sometimes also called *real-world spelling error correction* — in which errors may be corrected based on surrounding contextual word information. Many spelling error correction models have been suggested in the literature, among them, and most famously, the (generative) noisy channel model (Brill and Moore, 2000), discriminative models (Okazaki et al., 2008), finite-state techniques, as well as a plethora of local improvements and refinements for each class of models. In this work, rather than primarily suggesting new models for spelling error correction, we compare *general string-to-string translation* models developed in NLP contexts – typically, however, not within the area of spelling error correction — and survey methods for combining the outputs of the systems. The models we investigate have the following characteristics:

- They are *character-level*, that is, corrections are learned and implemented at the character-level, ignoring contextual words. Accordingly, in this work, our main focus is on isolated word-error correction, which may be considered harder than the context-sensitive spelling correction problem since surrounding contextual word cues are not available.[1] However, our experiments also include a real-world error correction setting.
- The models we survey are *general* in that they are not restricted to the spelling error correction task but can also be applied to many problems which require string-to-string translations, such as grapheme-to-phoneme conversion, transliteration, lemmatization, and others.[2] We think that generality and transferability of a model (in conjunction with accuracy) are central criteria of its quality.
- The models are *learned from data*, and in particular, are trained on pairs of strings of the form $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ is a misspelled word and $\mathbf{y}$ a desired correction.

The four approaches we survey are the Sequitur string-to-string translation model (Bisani and Ney, 2008), DirecTL+ (Jiampojamarn et al., 2010a), the contextual edit distance model suggested in Cotterell et al. (2014), and a model adaption of Eger (2012) which we call AliSeTra (Align-Segment-Translate). Although the first two models

[1] Also, one solution for real-world spelling error correction is to generate several candidates from an isolated spelling error correction model and then select the most likely candidate based on a word-level language model. In this sense, targeting isolated spelling error correction may be the first, and crucial, step in real-world spelling error correction.

[2] The only type of restrictions that our models make are *monotonicity* between input and output string characters, but otherwise allow, for instance, for many-to-many character relationships between input and output strings. This scenario is sometimes also referred to as *substring-to-substring* translation.

(and also the last) have been developed within the field of grapheme-to-phoneme (G2P) conversion and have been applied to related fields such as transliteration, too, their potential for the field of spelling error correction has apparently not yet been examined.[3]

We examine the suitability of the selected string-to-string translation models regarding the task of spelling error correction. To this end, we review the performance of these models and study the impact of additional resources (such as dictionaries and language models) on their effectiveness. Further, we investigate how to combine the output of the systems in order to get a system that performs as least as good as each of its component models. Note that combining string-valued variables is not a trivial problem since, for instance, the lengths of the strings predicted by the different systems may differ.

We show that by using k-best decoding in conjunction with a lexicon (dictionary), the string-to-string translation models considered here achieve much better results on the spelling correction task than three baselines, namely, edit distance, weighted edit distance and the Brill and Moore model. On two different data sets, three of the four models achieve word accuracy rates which are 5% resp. 25% better than the Brill and Moore baseline, which itself improves considerably upon edit distance and weighted edit distance. We also show that combining the models via language model weighted majority voting leads to yet another significant performance boost.

The article is structured as follows. In Section 2, we survey related work. In Section 3, we discuss the four string-to-string translation models and explain our techniques of combining them. Section 4 outlines the datasets used for evaluating these systems, viz., a Latin OCR dataset and a dataset of spelling errors in English tweets. Section 5, addresses three questions: (1) what are the accuracies of the four models on two different spelling correction data sets; (2) how can we improve the systems' performances by means of k-best output lists, language models and dictionaries; and (3) how well does the ensemble perform for different combination techniques — we consider weighted majority voting as well as center string ensembles. In Section 6, we touch upon the real-world spelling correction task, making use of our results in Section 5. In Section 7, we conclude.

2. Related Work

Brill and Moore (2000) suggest to solve the spelling error correction problem in the framework of the noisy channel model via maximizing the product of source model (language model) and the channel model for correcting a false input. Toutanova and Moore (2002) refine this model by integration of phonetic information. Cucerzan and Brill (2004) apply the noisy channel approach repeatedly, with the intent to cor-

[3]Similar investigations of G2P-inspired models for other tasks have been conducted, e.g., for lemmatization (Nicolai et al., 2015; Eger, 2015a).

rect more complex errors. More recent approaches to the spelling error correction problem include Okazaki et al. (2008), who suggest a discriminative model for candidate generation in spelling correction and, more generally, string transformation, and Wang et al. (2014), who propose an efficient log-linear model for correcting spelling errors, which, similar to the Brill and Moore (2000) model, is based on complex substring-to-substring substitutions. Farra et al. (2014) suggest a context-sensitive character-level spelling error correction model. Gubanov et al. (2014) improve the Cucerzan and Brill (2004) model by iterating the application of the basic noisy channel model for spelling correction in a stochastic manner.

Recently, there has been a surge of interest in solving the spelling error correction problem via the web (e.g., Whitelaw et al., 2009; Sun et al., 2010) and to correct query strings for search engines (e.g., Duan and Hsu, 2011, and many others). Further approaches to spelling correction include finite state techniques (e.g., Pirinen and Lindén, 2014) and deep graphical models (e.g., Raaijmakers, 2013). Kukich (1992) summarizes many of the earlier approaches to spell checking such as based on trie-based edit distances.

As mentioned, the models for spelling correction surveyed here are closely related to research on more general string-to-string transformation (translation) problems. This includes a variety of different models such as Cortes et al. (2005); Dreyer et al. (2008); Jiampojamarn et al. (2008); Bisani and Ney (2008); Cotterell et al. (2014); Wang et al. (2014); Sutskever et al. (2014); Novak et al. (2015).

3. Models

3.1. Alignment modeling

Two of the string-to-string translation systems evaluated below, DirecTL+ and AliSeTra, rely on *alignments* between input and output sequences $(\mathbf{x}, \mathbf{y})$. Since relationships between characters in spelling correction are typically of a complex nature as exemplified in Table 2, we assume that a *(monotone) many-to-many alignment* paradigm is the most suitable approach to modeling alignments in this scenario. We employ the monotone many-to-many aligner described in Jiampojamarn et al. (2007).[4] An implementation is available online at `https://code.google.com/p/m2m-aligner/`.

3.2. DirecTL+

DirecTL (Jiampojamarn et al., 2008, 2009) views string-to-string translation as a source sequence segmentation and subsequent sequence labeling task. The model extends its predecessor (Jiampojamarn et al., 2007) by folding the segmentation and

[4]This is an unsupervised many-to-many aligner. While supervised aligners are potentially more accurate (Eger, 2015b), the benefit of improved alignments for subsequent string transduction tasks is often marginal, particularly when training data is abundant.

tagging methods into a joint module. DirecTL+ (Jiampojamarn et al., 2010a) is a discriminative model for string-to-string translation that integrates joint n-gram features into DirecTL. The model has been applied in the context of grapheme-to-phoneme conversion (Jiampojamarn et al., 2010a) and in related domains such as transliteration (Jiampojamarn et al., 2010b). An online implementation is available at `https://code.google.com/p/directl-p/`.

3.3. Sequitur

Sequitur (Bisani and Ney, 2008) implements a joint n-gram model for string-to-string translation that, in the translation process from $\mathbf{x}$ to $\mathbf{y}$, uses n-gram probabilities over pairs of substrings of the input and output sequence ('joint multigrams'). Duan and Hsu (2011) use a joint-multigram modeling, very much in the spirit of Sequitur, for query-string correction for search engines. A downloadable version of Sequitur is available at `http://www-i6.informatik.rwth-aachen.de/web/Software/g2p.html`.

3.4. AliSeTra

We develop our own model for string-to-string translation that, similarly to DirecTL+, treats string transduction as a sequence segmentation and subsequent sequence labeling task. In this approach, at *training time*, a sequence labeling model (in our case a discriminative conditional random field) is trained on many-to-many aligned data. Simultaneously, a sequence labeling module is trained for segmenting input sequences by ignoring the segmented $\mathbf{y}$ sequences in the aligned data, simply considering the segmented $\mathbf{x}$ sequences. We use a binary encoding scheme similarly as in Bartlett et al. (2008) and Eger (2013) for learning sequence segmentation. At *test time*, an input string $\mathbf{x}$ is segmented via the segmentation module and then the sequence labeling model is applied to obtain the output sequence. In contrast to DirecTL+, this approach ignores joint n-gram features and resorts to the pipeline approach to string-to-string translation. Its benefit is that it may be used in conjunction with any state-of-the-art sequence labeling system, so it may directly profit from improvements in tagging technology. We use CRF++ as a sequence labeler.[5] We call this model AliSeTra (Align-Segment-Translate). In Table 1, we illustrate its decoding phase and show sample aligned training data on which the sequence labeling models in AliSeTra are trained.

3.5. Contextual Edit Distance

Cotterell et al. (2014) design a discriminative string-to-string translation model where $p(\mathbf{y}|\mathbf{x})$ is modeled via a probabilistic finite state transducer that encodes weighted edit operations transforming an input string $\mathbf{x}$ into an output string $\mathbf{y}$ (weighted

[5] Downloadable from `https://code.google.com/p/crfpp/`.

li-a-b-i-t-o	h-a-b-i-t-o	adliuc	$\rightsquigarrow$	a-d-li-u-c
a-d-j-u-t-o-r-i-u-ni	a-d-j-u-t-o-r-i-u-m			$\downarrow\downarrow\downarrow\downarrow\downarrow$
p-c-r-c-e-p-i-t	p-e-r-c-e-p-i-t			a-d-h-u-c

Table 1. Latin OCR spelling errors and their corrections. Left: Sample monotone many-to-many aligned training data, as obtained from the alignment procedure discussed in text. Alignment of characters indicated by dashes ('-')), one alignment per line. Right: AliSeTra *at test time. A new input string,* adliuc*, is first segmented into* a-d-li-u-c*, via a segmentation module trained on the segmented* $\mathbf{x}$ *strings in the training data. Then a tagging model, trained on the monotone many-to-many aligned pairs of* $(\mathbf{x}, \mathbf{y})$ *strings, assigns each (multi-)character in the segmentation its label, which can be a character or a multicharacter. This yields the predicted correction* adhuc *('hitherto').*

edit distance). Moreover, in their design, edit operations may be conditioned upon input and output context,[6] thus leading to a *stochastic contextual edit distance* model. An implementation is available from `http://hubal.cs.jhu.edu/personal/`.[7]

3.6. Baseline methods

As baseline methods for comparison, we use

- *edit distance* with the operations of insertion, deletion, and substitution as well as swapping of adjacent characters. That is, for a falsely spelled input $\mathbf{x}$, this measure determines the string $\mathbf{y}$ in a dictionary whose edit distance to $\mathbf{x}$ is lowest;
- *weighted edit distance*, in which the weight of edit operations is learned from data (we use the above named many-to-many aligner with edit operations restricted appropriately to induce training sets) rather than set exogenously;[8]
- and the Brill and Moore model (Brill and Moore, 2000), which embeds a substring-to-substring translation model into a generative noisy channel framework. In this, the channel probability $p(\mathbf{x}|\mathbf{y})$ is determined via (maximizing over) unigram models on substring segmentations of the form $\prod_i p(\mathbf{x}_i|\mathbf{y}_i)$, whereby

[6] The context is the preceding and subsequent characters in a string, not, e.g., the preceding words.

[7] The contextual edit distance model as designed in (Cotterell et al., 2014) is a locally normalized model suffering from the "label bias" problem and thus, potentially inadequate for our task. Although it has been primarily designed for incorporation in a Bayesian network over string-valued variables (Cotterell et al., 2015), we nonetheless include it here for comparison.

[8] In addition, we weight suggestions $\hat{\mathbf{y}}$, for an input $\mathbf{x}$, by a unigram word-level language model, which improves performance, as we found. The language model is trained on the same data sets as the language model for the Brill and Moore (2000) model; see below.

$x_1 \cdots x_r$ and $y_1 \cdots y_r$ are joint segmentations (i.e., an alignment) of x and y.[9] For the Brill and Moore (2000) model, we employ unigram word-level language models as source models.[10]

All these baselines are *dictionary-based*, that is, they retrieve corrections y given in a predefined dictionary D, which is typically advantageous (see our discussion below), but may lead to errors in case of, e.g., low quality of D. For efficient decoding, we employ a *trie*-based search strategy for finding corrections y in all three baseline methods presented. For edit distance, in case of ties between corrections — distinct forms y with same edit distance to x — we choose the lexicographically smallest form as the suggested correction.

For the English spelling error data (see below), we use the freely available (rule-based) spell checker Hunspell[11] as a reference.

3.7. System combination

Since we investigate multiple systems for spelling correction, a natural question to ask is how the outputs of the different systems can be combined. Clearly, this is a challenging task, and different approaches, with different levels of sophistication, have been suggested, both within the domain of machine translation (Rosti et al., 2007) and the field of string transductions (see, e.g., Cortes et al. (2014) for a survey). In this work, where the main goal is the comparison of existing approaches, we resort to *simple* combination techniques illustrated below. For an input string x — a wrongly spelled or a wrongly OCR recognized word form — let $y_1, \ldots, y_M$ denote the M predictions suggested by M different spelling correction systems. Then, we consider the following combination techniques:

- **Majority voting** chooses the most frequently suggested correction y among $y_1, \ldots, y_M$.
- **Weighted majority voting**: here, each suggested correction y_ℓ receives a weight $w_\ell \in \mathbb{R}$, and the correction y among $y_1, \ldots, y_M$ which maximizes $\sum_{\ell=1}^{M} w_\ell \mathbb{1}_{y_\ell = y}$ is chosen, where $\mathbb{1}_{a=b} = 1$ if $a = b$ and $\mathbb{1}_{a=b} = 0$ otherwise. We consider two weighting schemes:
  - *Accuracy weighted majority voting*: In this scheme, string y_ℓ receives weight w_ℓ proportional to the accuracy of system ℓ (e.g., as measured on a development set).

[9] In contrast, in SEQUITUR, for example, general n-*gram* models — rather than unigram models — over (x_i, y_i) pairs are used for modeling (joint) probabilities $p(x, y)$, indicating why SEQUITUR should typically outperform the Brill and Moore (2000) approach.

[10] For the Latin OCR data, as explicated below, these are trained on the Patrologia Latina (Migne, 1844–1855), and for the English Twitter data, the language model is based on a Wikipedia dump from 2013-09-04.

[11] `http://hunspell.sf.net`.

  - *Language model weighted majority voting*: In this scheme, suggestion $\mathbf{y}_\ell$ receives weight w_ℓ proportional to the language model likelihood of string $\mathbf{y}_\ell$.
- **Center string decoding**: We define the center string among $\mathbf{y}_1, \ldots, \mathbf{y}_M$, as the string $\mathbf{y} \in Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$ whose average edit distance to all other strings in Y is minimized (Gusfield, 1997). A center string can be seen as an (efficient) approximation to the concept of a *consensus string* (Gusfield, 1997), which does not need to be in Y.

Clearly, a drawback of all our suggested combination techniques is that they can only select strings $\mathbf{y}$ that belong to $\{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$. Hence, if none of the strings $\mathbf{y}_1, \ldots, \mathbf{y}_M$ is the true correction of the wrongly spelled form $\mathbf{x}$, then the system combination prediction will also be wrong. A strength of our combination techniques is that they are easily and efficiently implementable and interpretable.

4. Data

We conduct experiments on two data sets. The first is a Latin OCR spelling correction data set, which we obtained by comparison of an OCR scan of a subpart of the Patrologia Latina (Migne, 1844–1855) with the original in electronic form. The second is a data set of spelling errors in tweets,[12] which we refer to as Twitter data set. For the Latin data, we automatically extracted pairs of strings $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ denotes a wrongly recognized/spelled OCR form and $\mathbf{y}$ its desired correction, via the Unix shell command `diff`, applied to the original text and its OCR scan. This yielded about 12,000 pairs of $(\mathbf{x}, \mathbf{y})$ strings. From this, we excluded all string pairs containing upper case or non-ASCII characters, as some of our systems could only deal with lower-case ASCII characters. This resulted in a much smaller (and cleaner) data set comprising 5,213 string pairs. For the Twitter data, we took the first 5,000 word pairs of the respective data set for testing and training. We removed two word pairs which contained underscores in the $\mathbf{x}$ strings, for the same reason as indicated above.

Table 2 illustrates some of the relationships between characters (or character subsequences) in Latin and English strings and their spelling corrections. As is well-known, in the field of classical spelling correction, as the Twitter dataset represents, errors are often driven by 'phonetic similarity' of characters representing sounds, such as *a/e*, *u/ou*, etc., or keyboard adjacency of the characters in question such as *n/m*, *c/v*, etc. In contrast, OCR spelling errors typically derive from the *visual* similarity of characters, such as *li/h*, *n/ra*, *t/l*, *i/j*, *in/m*, etc. As Table 2 also illustrates, more complex many-to-many relationships between characters of $(\mathbf{x}, \mathbf{y})$ pairs may not be uncommon; and they allow for a seemingly plausible interpretation of the processes underlying string transformations. For example, it seems plausible to assume that an OCR system mis-

[12] Available from `http://luululu.com/tweet/`.

takes *h* for *li*, rather than assuming that, for instance, it confuses *h* with *l* and subsequently inserts an *i*.

n → ra	p**n**eterea ⟶ p**ra**eterea	mm → m	co**mm**ing ⟶ co**m**ing
li → h	ad**li**uc ⟶ ad**h**uc	n → m	victi**n** ⟶ victi**m**
i → j	**i**uventam ⟶ **j**uventam	t → th	**t**ink ⟶ **th**ink
t → l	il**t**ustri ⟶ il**l**ustri	u → ou	w**u**ld ⟶ w**ou**ld
in → m	**in**isero ⟶ **m**isero	a → e	emerg**a**ncy ⟶ emerg**e**ncy
c → e	quoqu**c** ⟶ quoqu**e**	c → v	ha**c**e ⟶ ha**v**e

Table 2. Sample substring substitution patterns in Latin OCR data (left) and English Twitter data (right), indicated and in bold. The patterns were found via automatic alignment of string pairs.

5. Isolated error correction

System parametrizations

We run the four systems of Section 3 using the following parametrizations. For SEQUITUR, we train 7 successive models, where parameters are, in each case, optimized on a heldout 5% development set. For ALISETRA, we set the C constant in the CRF++ implementation, which determines over-/underfitting of the model, to the default value of 1. For k-best decoding, we employ a '$k_1 \times k_2$ strategy' for ALISETRA:[13] at test time, each string is segmented into the k_1 most likely segmentations, and then the sequence labeling model — we take as features all sequence m-grams that fit inside a window of size 5 centered around the current position in the segmented string — transduces each of these into the k_2 most likely corrected strings. Thus, this yields $k_1 \times k_2$ output string suggestions; we multiply the segmentation probabilities with the transduction probabilities to obtain an overall probability of a corrected string. Then, we re-sort the obtained corrections and keep the k most likely. For the DIRECTL+ model, we choose, as context features, all m-grams inside a window of size 5 around the current position, as in the ALISETRA setting; we train a linear chain of order 1, set the joint multigram switch to 3 and the joint forward multigram switch to 1 (increasing the last three parameters did not seem to lead to better results, but only to longer runtimes). For CONTEXTUAL EDIT DISTANCE, we choose the best-performing $(1, 1, 1)$ topology from the Cotterell et al. (2014) paper, which considers as context the previous and next x string characters and the previous y string character (the value of the backoff parameter is 0.5). In terms of training times on a 2.54 GHz processor, train-

[13] In all experiments, we set $k_1 = 5$ and $k_2 = 50$.

ing the first three models ran in several hours, across all folds, while the Contextual Edit Distance model took days to train.

Evaluation setup

For the evaluation of our results, we employ 10-fold repeated random subsampling validation, in which, for each fold, we randomly split the data sets into training vs. test sets of size 90% vs. 10% of the whole data. Note that in random subsampling validation, training (as well as test) sets may overlap, across different folds.

Below, we indicate the performance of each of the four general string-to-string translation systems outlined in Section 3 in two different settings. In the *first setting*, we simply check whether the first-best string $\hat{\mathbf{y}}$ predicted by a system S for an input string $\mathbf{x}$ matches $\mathbf{y}$, the true correction for input string $\mathbf{x}$. This is the typical evaluation scenario, e.g., in grapheme-to-phoneme conversion and related string-to-string translation fields such as transliteration. In an *alternative setting*, we let each system emit its k-best output predictions for an input string $\mathbf{x}$, in decreasing order of (system-internal) probability, and then choose, as the system's prediction for $\mathbf{x}$, the first-best string $\mathbf{y}^j$, for $j = 1, \ldots, k$, *that occurs in a predefined dictionary* D. If no string $\mathbf{y}^1, \ldots, \mathbf{y}^k$ is in D, we choose $\mathbf{y}^1$ as the system's prediction, as in the standard setting. Note that our first setting is a special case of the second setting in which $k = 1$.

Consulting a dictionary is done by most approaches to spelling correction. Combining a dictionary with k-best decoding in the manner described is apparently a plausible solution to integrating a dictionary in the setup of general string-to-string translation models. Note that our approach allows for predicting output strings that are not in the dictionary, which may be advantageous in case of low dictionary quality — but even if the quality of the dictionary is good, desired output strings may be missing (cf. Table 3).

For Latin, we choose a subset of ColLex.LA (Mehler et al., 2015) as our dictionary of choice and for English, we use ColLex.EN (vor der Brück et al., 2014).[14] Table 3 gives the number of entries in both lexicons as well as OOV numbers.

	Number of unique entries	OOV rate
Subset of ColLex.LA	4,269,104	$57/5213 = 1.09\%$
ColLex.EN	3,998,576	$189/4998 = 3.78\%$

Table 3. Dictionaries, their sizes, and OOV rates (number of corrections in each data set not in the dictionary).

[14] Both dictionaries are available from `http://collex.hucompute.org/`.

In both settings, we use **word accuracy** (WACC) as a performance measure, defined as the number of correctly translated strings over the total number of translated strings,

$$\text{WACC} = \frac{\sum_{i=1}^{n} \mathbb{1}_{\hat{y}_i = y_i \mid x_i}}{n},$$

where n is the size of the test set and $\mathbb{1}_{\hat{y}_i = y_i \mid x_i}$ is one or zero, depending on whether $\hat{y}_i = y_i$ or not (we use the $\mid$ notation to indicate dependence of $y_i/\hat{y}_i$ on input x_i).[15]

5.1. Individual system results

Tables 4 and 5 list the results for the two data sets when using our above dictionary-based strategy with 1-best and 80-best decoding. Clearly, 80-best decoding yields much better results for all of the four methods, where word accuracy increases from about $16 - 70\%$ on the Latin OCR and $5 - 30\%$ on the Twitter data, relative to 1-best decoding, across all systems. This confirms that a dictionary may be very helpful in (OCR) spelling correction and that simple k-best decoding and first-best dictionary selection can be a good solution for integrating a dictionary into general string-to-string translation systems. In Figures 1 and 2, we plot each system's performance as a function of k in the k-best decoding strategy.

We also note that three of the four systems introduced in Section 3 — namely, AliSeTra, DirecTL+, Sequitur — have a very similar performance across the two data sets, whereas Contextual Edit Distance performs much worse, particularly in 1-best decoding. We attribute this to the fact that contextual edit distance considers much less context in our setup than do the other three systems.[16] Moreover, it operates on a single-character, rather than on a substring, or multi-character, level, which further reduces its contextual awareness.[17] However, we see that differences in system performances decrease as k increases. For example, for $k = 1$, the best system is approximately 60%/57% better than Contextual Edit Distance on the Latin OCR/Twitter data sets — while for $k = 80$, this reduces to 9%/28%. This indicates that Contextual Edit Distance may enumerate many of the relevant correct strings, for given input strings x, but has a higher chance of erring in correctly ranking them. We also note that the Twitter data set is apparently harder than the Latin OCR data set, as all systems exhibit worse performance on the former data set. This is, among other things,

[15] When an input x has multiple distinct translations in the test data set — e.g., *tis* $\longrightarrow$ *this, is, its* — then, in the evaluation, we randomly choose one of these translations as the true translation. As discussed below, such cases happen relatively rarely. For example, in the Latin OCR data, 88.5% of all x forms have a unique correction associated with them, while in the Twitter data, this number is 61.5%.

[16] Increasing context size is critical, as the program's runtime is excessive. We did not experiment with larger context sizes for Contextual Edit Distance.

[17] Finally, contextual edit distance is locally normalized and thus suffers from the label bias problem as discussed earlier.

due to the fact that the Twitter data is generally more ambiguous than the Latin data in that an input string $\mathbf{x}$ is potentially related to more candidate alternatives.[18]

Model	1-best	80-best
AliSeTra	74.66 ± 1.26	87.33 ± 1.26
DirecTL+	$\mathbf{75.95 \pm 1.65}$	$\mathbf{88.35 \pm 1.54}$
Sequitur	73.67 ± 1.85	87.44 ± 1.90
Contextual Edit Distance	47.55 ± 1.77	81.12 ± 1.28
Edit distance		45.30 ± 2.04
Weighted edit distance		73.67 ± 1.21
Brill and Moore		84.20 ± 2.23

Table 4. Latin OCR data: Word accuracy in % for the k*-best decoding strategy explicated in the text, and comparison with baseline methods; note, in particular, that we use a dictionary in conjunction with* k*-best decoding (*1*-best decoding is tantamount to ignoring the dictionary). The baseline methods are dictionary-based by their design, so the numbers simply indicate their word accuracy for their first-best prediction. In bold: Statistically indistinguishable best results (paired* t*-test,* 5% *level).*

Model	1-best	80-best
AliSeTra	$\mathbf{68.38 \pm 1.52}$	$\mathbf{72.98 \pm 2.01}$
DirecTL+	$\mathbf{68.15 \pm 1.56}$	71.65 ± 2.12
Sequitur	63.01 ± 1.54	70.46 ± 1.60
Contextual Edit Distance	43.52 ± 2.28	56.78 ± 1.86
Edit distance		16.81 ± 1.78
Weighted edit distance		33.69 ± 2.11
Brill and Moore		58.08 ± 3.00
Hunspell		41.42 ± 1.96

Table 5. Twitter spelling correction data: Word accuracy in % for the k*-best decoding strategy explicated in the text, and comparison with baseline methods.*

[18] In the Latin OCR data, each $\mathbf{x}$ is on average associated with 1.0037 distinct $\mathbf{y}$ forms, while in the Twitter data, there are 1.1101 distinct $\mathbf{y}$ forms per $\mathbf{x}$ form. To illustrate, the possible corrections of *ot* in the Twitter data are *on, of, it, to, got, or, out*; similarly, *wat* may be corrected by *what, was, way, want, at*, etc. While in the evaluation, we remove this uncertainty by randomly assigning one of the strings as the correct output for a given input, at training time, this may lead to more inconsistency and ambiguity.

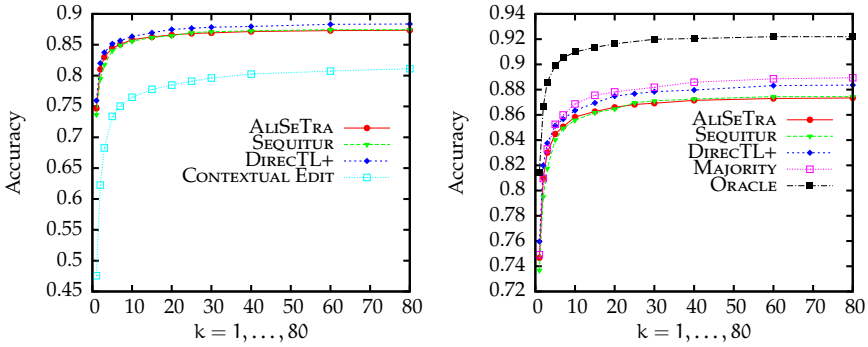

Figure 1. Latin OCR data, word accuracy as a function of k *in the* k*-best decoding strategy outlined in the text. Left: the four systems introduced in Section 3. Right: Three of the systems (excluding* Contextual Edit Distance*, for clarity) plus majority voting and oracle performance.*

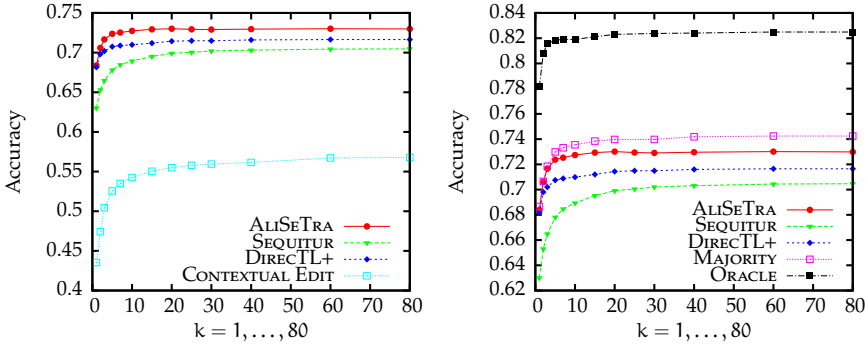

Figure 2. Twitter data, word accuracy as a function of k *in the* k*-best decoding strategy outlined in the text. Left: the four systems introduced in Section 3. Right: Three of the systems (excluding* Contextual Edit Distance*, for clarity) plus majority voting and oracle performance.*

Comparing the figures in the graphs and tables, we also see that three of the four general string-to-string translation systems surveyed perform much better than the baselines edit distance, weighted edit distance, and the Brill and Moore model. For instance, on the Latin OCR data, the best system is roughly 5% better than the performance of the Brill and Moore model, which itself is considerably better than edit

distance or weighted edit distance, while on the Twitter data, this difference amounts to more than 25%. Oftentimes, the three of the four general string-to-string translation systems also perform on a level close to or above the level of the compared baselines, *even without using a dictionary*, as the 1-best results indicate.

In Figure 3, we provide another measure of system performance, *recall-at-*k. Under this measure, a system is correct for an input $\mathbf{x}$ if the true correction $\mathbf{y}$ is among the system's k-best predictions $\mathbf{y}^1, \ldots, \mathbf{y}^k$. Clearly, for fixed k, each system's performance under this measure must be at least as good as under the previous word accuracy measure for the k-best decoding strategy. Recall-at-k may be an important indicator for real-world spell checking, which often relies on a candidate generation module and a ranker for the candidates. Then, it may be sufficient for the candidate generation module to *generate* the true correction, as long as the ranker (often a word level n-gram model) can adequately discriminate between alternatives.

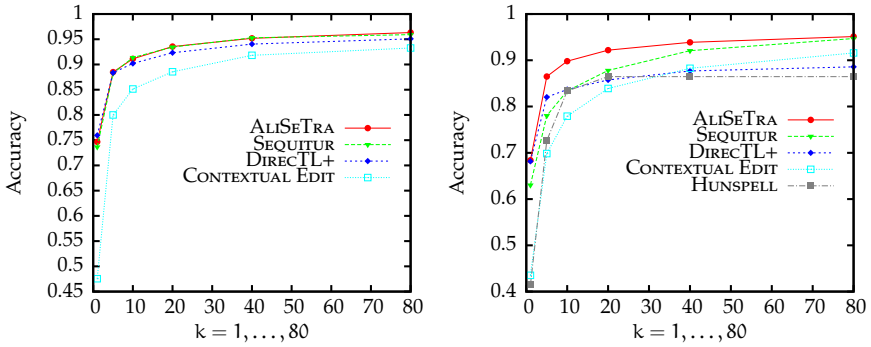

*Figure 3. Recall-at-*k *as described in the text. Left: Latin OCR data. Right: Twitter data.*

As seen in the figure, results are similar as in the previous setting — system performances increase significantly as k increases and system differences decrease in k. Interestingly, DirecTL+ appears to perform relatively worse under this measure than under the word accuracy measure, indicating that it seems to do a relatively better job in ranking alternatives, compared to the other systems. In contrast, Hunspell and Contextual Edit Distance, for example, which perform badly at predicting the exact true correction for an input, nonetheless appear relatively more capable of at least generating the true correction among their predictions. We also conclude that given that the recall-at-k of some of the systems is above 95% and 90% for the Latin OCR and Twitter data sets, respectively, while k-best decoding plus dictionary selection as outlined above yields word accuracy rates of (only) about 88% and 72%, respec-

tively, our presented dictionary k-best decoding strategy could in principle be much improved upon.

5.2. System combination results

In Tables 6 and 7, we show results for the different system combination techniques outlined in Section 3. For the four systems surveyed in this work, we use the 80-best dictionary decoding strategy as outlined above as a basis for the system combination. We see that majority voting, center string combination, and weighted majority voting can increase performance significantly over the individual system accuracies. Majority voting gives slightly better results than center string combination. Even including weaker systems can be beneficial as the tables show. Typically, best results are obtained via integration of all systems except for (individually quite poor) standard edit distance. Compared to the individual systems, majority voting increases word accuracy by as much as 2% on the Latin OCR data set and as much as 5% on the Twitter data set; performance increases for center string combination are 1.1% and 3.5%, respectively.

Latin OCR				
Models	**Majority** (MV)	**Center String**	**Acc-MV**	**LM-MV**
A+D+S	88.52 ± 1.47	88.60 ± 1.50	$88.62^* \pm 1.40$	90.99 ± 1.32
+CED	88.93 ± 1.51	88.89 ± 1.48	$89.12^* \pm 1.45$	91.46 ± 1.16
+BM	89.82 ± 1.55	89.62 ± 1.39	$89.76^* \pm 1.22$	93.13 ± 0.92
+WE	$\mathbf{90.16 \pm 1.22}$	$\mathbf{89.93 \pm 1.36}$	$\mathbf{90.07^* \pm 1.33}$	$\mathbf{93.33 \pm 0.97}$
+ED	89.74 ± 1.45	89.33 ± 1.38	$89.80^* \pm 1.30$	93.27 ± 0.90

Table 6. Word accuracies for system combination techniques on Latin OCR data. Systems abbreviated by their first letters or initials (WE is weighted edit distance, ED is standard edit distance). In each column: statistically indistinguishable best results, paired t*-test, 5% level. The results for accuracy-weighted majority voting are starred because we used the accuracies as obtained on the test data (usually, a development data set would need to be used for this), so that the results are 'upward' biased.*

Accuracy-weighted majority voting does not typically result in large improvements over simple majority voting, if at all. Conversely, when we train a 10-gram character level language model (for Latin, on the original text from which the spelling correction $(\mathbf{x}, \mathbf{y})$ pairs were obtained; for Twitter, on the remaining roughly $35,000$ $\mathbf{y}$ strings that were not used in training/testing), and perform language model weighted majority voting, then this significantly increases results, by 3.5% on the Latin OCR data and 4.6% on the Twitter data, over standard majority voting combination.

English Twitter					
Models	**Maj.** (MV)	**Center Str.**	**Acc-MV**	**LM$_{\text{Twitter}}$-MV**	**LM$_{\text{Europarl}}$-MV**
A+D+S	74.56 ± 1.90	**75.08 ± 2.05**	74.87* ± 2.05	77.80 ± 1.59	**76.34 ± 1.51**
+CED	74.34 ± 2.24	**75.06 ± 2.13**	75.03* ± 2.47	78.28 ± 1.87	75.23 ± 1.62
+BM	76.09 ± 2.06	**75.23 ± 2.31**	76.09* ± 2.38	80.11 ± 1.93	74.20 ± 2.14
+WE	**76.69 ± 1.79**	**75.57 ± 2.23**	**76.69*** ± 2.10	**80.58 ± 1.94**	72.49 ± 1.83
+ED	75.49 ± 1.89	74.31 ± 2.02	**76.69*** ± 2.11	**80.69 ± 1.98**	72.56 ± 1.95

Table 7. Word accuracies for system combination techniques on English Twitter data.

Note that a language model may lead to deteriorations in results if being trained on data very dissimilar to the data on which it is to be applied and when weak systems are integrated into the majority voting process. For example, when we train a 10-gram character level language model on the English part of the Europarl corpus (Koehn, 2005), then language model weighted majority voting with 7 systems almost drops down to the word accuracy level of the single best system in the ensemble.

6. Real-world error correction

Finally, we consider the real-world spelling correction problem in our context, focusing on the Latin OCR data. To this end, we train two components: a spelling error correction model as outlined in the previous section and a language model (LM). We train the two most successful spelling correction systems from our previous setup — DIRECTL+ and ALISETRA — on the previously described Latin OCR data,[19] this time not excluding word pairs containing upper-case or non-ASCII characters (so as to provide a 'real-world' situation). In addition, we train a 5-gram Kneser-Ney word-level LM via the SRILM toolkit[20] (Stolcke, 2002) on the union of the Patrologia Latina and the Latin Wikipedia data.[21] To combine the predictions of the LM and the discriminative string transducers, we opt for a power mean combination. In particular, for a potentially incorrect form $\mathbf{x}$, we let the respective OCR post-corrector output its K-best (here, $K = 80$) suggestions $\mathbf{y}_1, \ldots, \mathbf{y}_K$. For the LM, we score each of these suggestions $\mathbf{y}$ by querying the LM on the sequence $\mathbf{x}_{t-4} \cdots \mathbf{x}_{t-1}\mathbf{y}$, where $\mathbf{x}_{t-s}$ denotes the s-th word before word $\mathbf{x}$ at position t. Then, we choose the form $\hat{\mathbf{y}}$ as the suggested correction which maximizes

$$\text{PM}\left(\text{lm-score}(\mathbf{x}_{t-4} \cdots \mathbf{x}_{t-1}\hat{\mathbf{y}}), \text{tm-score}(\hat{\mathbf{y}}|\mathbf{x}); w_{\text{LM}}, 1 - w_{\text{LM}}, p\right)$$

[19] We keep 90% for training and 10% for testing.

[20] http://www.speech.sri.com/projects/srilm/

[21] Dump from 2015-10-10.

where lm-score denotes the LM score and tm-score denotes the score of the respective OCR transducer model. We normalize the scores such that they sum to 1 for all suggestions in the candidate list. Finally, $\mathrm{PM}(x, y; w_x, w_y, p)$ is the power mean $(w_x x^p + w_y y^p)^{1/p}$ where $w_x, w_y \geq 0$ with $w_x + w_y = 1$, and $p \in \mathbb{R}$. We consider here $p = 1$ (weighted arithmetic mean) and $p \to 0$ (weighted geometric mean); we refer to the latter case as $p = 0$, for convenience.

We consider two 'treatments', one in which we filter out suggestions $\hat{y}$ not in the lexicon, and one in which no such filtering takes place. We consider a form x as potentially incorrect only if x is not in our Latin lexicon. When comparing the post-corrected text with the original, we face the problem that the two texts are not identical in that the original, e.g., contains additional text such as insertions introduced by the texts' editors ('[0026A]'). Thus, we find it easiest to measure the improvement between the scanned text version and our post-correction by applying the Unix `diff` command to the two files.[22]

Table 8 shows the results, for different values of w_{LM} and $p = 0, 1$. We note some general trends: using geometric averaging is always better than using arithmetic averaging, and using the DirecTL+ corrector is usually better than using AliSeTra, which is in accordance with the results highlighted in Table 4. Moreover, making the LM weight too large is typically detrimental; in these experiments, values $\leq 1/2$ were found to be best, indicating that the post-correctors typically perform better than the LM. Finally, using the lexicon as a filtering device has been beneficial in 8 out of 20 cases, but led to worse results in the remaining cases. A possible explanation is that, after filtering suggestions by whether they are contained in the lexicon, the candidates' LM and OCR corrector scores change since we renormalize them. Hence, if for example the LM has attributed a high score to an incorrect form this score may become even higher after filtering, thus leading to higher probability of a wrong selection. Finally, we note that the `diff` measure value between the original text and its scan is 1794, so our post-correction improves this value by roughly 28% (1294 and 1302 for AliSeTra and DirecTL+, respectively, in the best settings). While this seems to be a moderate improvement, we note that many wrongly scanned forms are in our lexicon; in particular, this concerned ligatures such as æ in the scan *memoriæ* of *memoriae*. Hence, these forms were not corrected at all since our correction addressed only forms not available in our lexicon.

7. Conclusion

We considered the isolated spelling error correction problem as a specific subproblem of the more general string-to-string translation problem. In this respect, we investigated four general string-to-string transformation models that have been suggested

[22] To be precise, our command for comparing the two versions is
`diff post-corrected.txt orig.txt -y |grep "|\|<\|>"|wc -l`.

	Lexicon	OCR corrector	0	1/4	1/2	3/4	1
p = 1	+	AliSeTra	1389	1376	**1366**	1439	1504
p = 0	+	AliSeTra	1389	1361	**1356**	1401	1504
p = 1	-	AliSeTra	1451	1390	**1339**	1407	1475
p = 0	-	AliSeTra	1451	1316	**1294**	1357	1475
p = 1	+	DirecTL+	1343	1336	**1330**	1406	1466
p = 0	+	DirecTL+	1343	**1325**	1330	1344	1466
p = 1	-	DirecTL+	1417	**1343**	1356	1412	1449
p = 0	-	DirecTL+	1417	1314	**1302**	1315	1449

Table 8. Real-world OCR post-correction results as described in text. Different parametrizations and LM weights w_{LM}. Lower `diff` scores are better. In bold: best results in each row.

in recent years and applied them within the spelling error correction paradigm. Moreover, we investigated how a simple 'k-best decoding plus dictionary lookup' strategy performs in this context. We showed that such an approach can significantly outdo baselines such as the edit distance, weighted edit distance, and the noisy channel Brill and Moore model (Brill and Moore, 2000) applied for spelling error correction. In particular, we saw that in the named dictionary-based modus, (three of) the models surveyed here are much better than the baselines in ranking a set of candidate suggestions for a falsely spelled input. We have also shown that by combining the four models surveyed (and the baselines) via simple combination techniques, even better results can be obtained. Finally, we conducted real-world OCR correction experiments based on our trained systems and language models. The data and the dictionaries can be accessed via `https://www.hucompute.org/ressourcen/corpora` so that our findings may be used as a starting point for related research.

In future work, we intend to investigate more sophisticated combination techniques for combining outputs of several spell checkers, e.g., on the character-level, as done in Cortes et al. (2014); Eger (2015d,c); Yao and Kondrak (2015). We also intend to evaluate neural-network based techniques in the present scenario (Sutskever et al., 2014; Yao and Zweig, 2015). Finally, we plan to substitute the CRF++ tagger used in AliSeTra by a higher-order CRF tagger as described by Müller et al. (2013).

Acknowledgments

We gratefully acknowledge financial support by the BMBF via the research project *CompHistSem* (`http://comphistsem.org/home.html`). We also thank Tim Geelhaar and Roland Scheel for providing the OCR scan of the subpart of the Patrologia Latina on which our Latin OCR experiments are based.

Bibliography

Bartlett, Susan, Grzegorz Kondrak, and Colin Cherry. Automatic Syllabification with Structured SVMs for Letter-to-Phoneme Conversion. In McKeown, Kathleen, Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, editors, *ACL*, pages 568–576. The Association for Computational Linguistics, 2008. ISBN 978-1-932432-04-6. URL http://dblp.uni-trier.de/db/conf/acl/acl2008.html#BartlettKC08.

Bisani, Maximilian and Hermann Ney. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451, 2008. URL http://dblp.uni-trier.de/db/journals/speech/speech50.html#BisaniN08.

Brill, Eric and Robert C. Moore. An Improved Error Model for Noisy Channel Spelling Correction. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL)*, ACL '00, pages 286–293, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. doi: 10.3115/1075218.1075255. URL http://dx.doi.org/10.3115/1075218.1075255.

Cortes, Corinna, Mehryar Mohri, and Jason Weston. A General Regression Technique for Learning Transductions. In *Proceedings of the 22Nd International Conference on Machine Learning*, Proceedings of the International Conference on Machine Learning (ICML), pages 153–160, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102371. URL http://doi.acm.org/10.1145/1102351.1102371.

Cortes, Corinna, Vitaly Kuznetsov, and Mehryar Mohri. Ensemble Methods for Structured Prediction. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014.

Cotterell, Ryan, Nanyun Peng, and Jason Eisner. Stochastic Contextual Edit Distance and Probabilistic FSTs. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Baltimore, June 2014. URL http://cs.jhu.edu/~jason/papers/#cotterell-peng-eisner-2014. 6 pages.

Cotterell, Ryan, Nanyun Peng, and Jason Eisner. Modeling Word Forms Using Latent Underlying Morphs and Phonology. *Transactions of the Association for Computational Linguistics*, 3:433–447, 2015. ISSN 2307-387X. URL https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/480.

Cucerzan, S. and E. Brill. Spelling Correction as an Iterative Process that Exploits the Collective Knowledge of Web Users. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2004.

Dreyer, Markus, Jason Smith, and Jason Eisner. Latent-Variable Modeling of String Transductions with Finite-State Methods. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1080–1089. ACL, 2008. URL http://dblp.uni-trier.de/db/conf/emnlp/emnlp2008.html#DreyerSE08.

Duan, Huizhong and Bo-June (Paul) Hsu. Online Spelling Correction for Query Completion. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 117–126, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0632-4. doi: 10.1145/1963405.1963425. URL http://doi.acm.org/10.1145/1963405.1963425.

Eger, Steffen. S-Restricted Monotone Alignments: Algorithm, Search Space, and Applications. In *Proceedings of the Conference on Computational Linguistics (COLING)*, pages 781–798, 2012.

Eger, Steffen. Sequence Segmentation by Enumeration: An Exploration. *Prague Bull. Math. Linguistics*, 100:113–132, 2013. URL http://dblp.uni-trier.de/db/journals/pbml/pbml100.html#Eger13.

Eger, Steffen. Designing and comparing G2P-type lemmatizers for a morphology-rich language. In *Fourth International Workshop on Systems and Frameworks for Computational Morphology*, pages 27–40. Springer International Publishing Switzerland, 2015a.

Eger, Steffen. Do we need bigram alignment models? On the effect of alignment quality on transduction accuracy in G2P. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1175–1185, Lisbon, Portugal, September 2015b. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1139.

Eger, Steffen. Improving G2P from wiktionary and other (web) resources. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 3340–3344, 2015c.

Eger, Steffen. Multiple Many-to-Many Sequence Alignment for Combining String-Valued Variables: A G2P Experiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 909–919, Beijing, China, July 2015d. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1088.

Farra, Noura, Nadi Tomeh, Alla Rozovskaya, and Nizar Habash. Generalized Character-Level Spelling Error Correction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–167, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P14/P14-2027.

Gubanov, Sergey, Irina Galinskaya, and Alexey Baytin. Improved Iterative Correction for Distant Spelling Errors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 168–173, 2014. URL http://aclweb.org/anthology/P/P14/P14-2028.pdf.

Gusfield, Dan. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology*. Cambridge University Press, 1997. ISBN 0-521-58519-8.

Jiampojamarn, Sittichai, Grzegorz Kondrak, and Tarek Sherif. Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379, Rochester, New York, April 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N07/N07-1047.

Jiampojamarn, Sittichai, Colin Cherry, and Grzegorz Kondrak. Joint Processing and Discriminative Training for Letter-to-Phoneme Conversion. In *Proceedings of ACL-08: HLT*, pages 905–913, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P08/P08-1103.

Jiampojamarn, Sittichai, Aditya Bhargava, Qing Dou, Kenneth Dwyer, and Grzegorz Kondrak. DirecTL: a Language Independent Approach to Transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 28–31, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W09/W09-3504.

Jiampojamarn, Sittichai, Colin Cherry, and Grzegorz Kondrak. Integrating Joint n-gram Features into a Discriminative Training Framework. In *Proceedings of HLT-NAACL*, pages 697–700. The Association for Computational Linguistics, 2010a. ISBN 978-1-932432-65-7. URL http://dblp.uni-trier.de/db/conf/naacl/naacl2010.html#JiampojamarnCK10.

Jiampojamarn, Sittichai, Kenneth Dwyer, Shane Bergsma, Aditya Bhargava, Qing Dou, Mi-Young Kim, and Grzegorz Kondrak. Transliteration Generation and Mining with Limited Training Resources. In *Proceedings of the 2010 Named Entities Workshop*, pages 39–47, Uppsala, Sweden, July 2010b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-2405.

Koehn, Philipp. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT, AAMT. URL http://mt-archive.info/MTS-2005-Koehn.pdf.

Kukich, Karen. Techniques for Automatically Correcting Words in Text. *ACM Comput. Surv.*, 24(4):377–439, Dec. 1992. ISSN 0360-0300. doi: 10.1145/146370.146380. URL http://doi.acm.org/10.1145/146370.146380.

Mehler, Alexander, Tim vor der Brück, Rüdiger Gleim, and Tim Geelhaar. Towards a Network Model of the Coreness of Texts: An Experiment in Classifying Latin Texts using the TTLab Latin Tagger. In Biemann, Chris and Alexander Mehler, editors, *Text Mining: From Ontology Learning to Automated Text Processing Applications*, Theory and Applications of Natural Language Processing, pages 87–112. Springer, Berlin/New York, 2015.

Migne, Jacques-Paul, editor. *Patrologiae cursus completus: Series latina*. 1–221. Chadwyck-Healey, Cambridge, 1844–1855.

Mitankin, Petar, Stefan Gerdjikov, and Stoyan Mihov. An Approach to Unsupervised Historical Text Normalisation. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, Proceedings of DATeCH '14, pages 29–34, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2588-2. doi: 10.1145/2595188.2595191. URL http://doi.acm.org/10.1145/2595188.2595191.

Müller, Thomas, Helmut Schmid, and Hinrich Schütze. Efficient Higher-Order CRFs for Morphological Tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D13-1032.

Nicolai, Garrett, Colin Cherry, and Grzegorz Kondrak. Inflection Generation as Discriminative String Transduction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 922–931, Denver, Colorado, May–June 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N15-1093.

Novak, Josef Robert, Nobuaki Minematsu, and Keikichi Hirose. Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the WFST framework. *Natural Language Engineering*, 2015.

Okazaki, Naoaki, Yoshimasa Tsuruoka, Sophia Ananiadou, and Jun'ichi Tsujii. A Discriminative Candidate Generator for String Transformations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, EMNLP '08, pages 447–456, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1613715.1613772.

Pirinen, Tommi A. and Krister Lindén. State-of-the-Art in Weighted Finite-State Spell-Checking. In *Computational Linguistics and Intelligent Text Processing - 15th International Conference, CICLing 2014, Kathmandu, Nepal, April 6-12, 2014, Proceedings, Part II*, pages 519–532, 2014. doi: 10.1007/978-3-642-54903-8_43. URL http://dx.doi.org/10.1007/978-3-642-54903-8_43.

Raaijmakers, Stephan. A deep graphical model for spelling correction. In *Proceedings BNAIC 2013*, 2013.

Reynolds, L. D. and Nigel Wilson. *Scribes and scholars. A guide to the transmission of Greek and Latin literature*. Clarendon Press, Oxford, 3. aufl. edition, 1991. ISBN 0-19-872145-5.

Rosti, Antti-Veikko I., Necip Fazil Ayan, Bing Xiang, Spyridon Matsoukas, Richard M. Schwartz, and Bonnie J. Dorr. Combining Outputs from Multiple Machine Translation Systems. In Sidner, Candace L., Tanja Schultz, Matthew Stone, and ChengXiang Zhai, editors, *Proceedings of HLT-NAACL*, pages 228–235. The Association for Computational Linguistics, 2007. URL http://dblp.uni-trier.de/db/conf/naacl/naacl2007.html#RostiAXMSD07.

Springmann, Uwe, Dietmar Najock, Hermann Morgenroth, Helmut Schmid, Annette Gotscharek, and Florian Fink. OCR of historical printings of Latin texts: problems, prospects, progress. In *Digital Access to Textual Cultural Heritage 2014, DATeCH 2014, Madrid, Spain, May 19-20, 2014*, pages 71–75, 2014. doi: 10.1145/2595188.2595205.

Stolcke, Andreas. SRILM-an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286, November 2002.

Sun, Xu, Jianfeng Gao, Daniel Micol, and Chris Quirk. Learning Phrase-Based Spelling Error Models from Clickthrough Data. In Hajic, Jan, Sandra Carberry, and Stephen Clark, editors, *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 266–274. The Association for Computational Linguistics, 2010. ISBN 978-1-932432-67-1. URL http://dblp.uni-trier.de/db/conf/acl/acl2010.html#SunGMQ10.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014.

Toutanova, Kristina and Robert C. Moore. Pronunciation Modeling for Improved Spelling Correction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 144–151. ACL, 2002. URL http://dblp.uni-trier.de/db/conf/acl/acl2002.html#ToutanovaM02.

vor der Brück, Tim, Alexander Mehler, and Md. Zahurul Islam. ColLex.EN: Automatically Generating and Evaluating a Full-form Lexicon for English. In *Proceedings of LREC 2014*, Reykjavik, Iceland, 2014.

Wang, Ziqi, Gu Xu, Hang Li, and Ming Zhang. A Probabilistic Approach to String Transformation. *IEEE Trans. Knowl. Data Eng.*, 26(5):1063–1075, 2014. doi: 10.1109/TKDE.2013.11. URL http://doi.ieeecomputersociety.org/10.1109/TKDE.2013.11.

Whitelaw, Casey, Ben Hutchinson, Grace Y. Chung, and Gerard Ellis. Using the Web for Language Independent Spellchecking and Autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 890–899, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-62-6. URL http://dl.acm.org/citation.cfm?id=1699571.1699629.

Yao, Kaisheng and Geoffrey Zweig. Sequence-to-Sequence Neural Net Models for Grapheme-to-Phoneme Conversion. *CoRR*, abs/1506.00196, 2015.

Yao, Lei and Grzegorz Kondrak. Joint Generation of Transliterations from Multiple Representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 943–952, Denver, Colorado, May–June 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N15-1095.

Address for correspondence:
Steffen Eger
eger@ukp.informatik.tu-darmstadt.de
Technische Universität Darmstadt
Hochschulstraße 10, 64289 Darmstadt, Germany