Unsupervised Text Segmentation for Automated Error Reduction

Furrer, Lenz

Abstract: Challenging the assumption that traditional whitespace/punctuation-based tokenisation is the best solution for any NLP application, I propose an alternative approach to segmenting text into processable units. The proposed approach is nearly knowledge-free, in that it does not rely on language-dependent, man-made resources. The text segmentation approach is applied to the task of automated error reduction in texts with high noise. The results are compared to conventional tokenisation.

Originally published at:
Furrer, Lenz (2014). Unsupervised Text Segmentation for Automated Error Reduction. In: KONVENS 2014, Hildesheim, 8 October 2014 - 10 October 2014. Universität Hildesheim, 178-185.

Unsupervised Text Segmentation for Automated Error Reduction

Lenz Furrer
University of Zurich
Binzmühlestr. 14, CH-8050 Zürich
lenz.furrer@uzh.ch

Abstract

Challenging the assumption that traditional whitespace/punctuation-based tokenisation is the best solution for any NLP application, I propose an alternative approach to segmenting text into processable units. The proposed approach is nearly knowledge-free, in that it does not rely on language-dependent, manmade resources. The text segmentation approach is applied to the task of automated error reduction in texts with high noise. The results are compared to conventional tokenisation.

1 Introduction

Dividing written text into small units is one of the most basic and fundamental steps in Natural Language Processing (NLP). Generally, this task does not attract much attention, as it is most often carried out by relying on a language's orthography for marking word boundaries. Whitespace/punctuation-based tokenisation – which is applicable to most mainstream languages covered in NLP literature – is not necessarily the optimal starting point for every NLP application. In this work, I investigate the use of an alternative segmentation approach by applying it to the task of automatically reducing errors in documents of amendable text quality.

The experiments reported in this work were carried out with electronic documents created by Optical Character Recognition (OCR) software. Modern OCR tools provide a high recognition accuracy. However, it is often desirable to improve the text quality of OCR-generated documents in a post-processing step, especially if they are in a challenging format such as historical books.

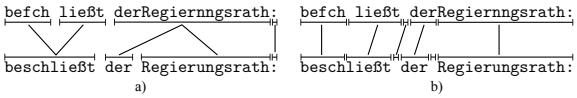

Figure 1: A line of OCRed text (upper line) and its correction (lower line), segmented with traditional, whitespace-based tokenisation (a) and with a non-standard segmentation method (b). Note that the space character is often, but not always, treated as a separate token in (b). The line can approximately be translated as 'the executive council decides'.

Most post-correction attempts operate on the level of words, i. e. tokenised text. However, recognition errors can lead to erroneous tokenisation if word separators are omitted or falsely inserted, as shown in figure 1. Segmentation errors need special attention in word-based correction approaches. As illustrated in example (a), tokenisation is misled by missing or falsely inserted space characters, producing disrupted and run-on tokens which cannot be corrected by comparing words one by one. An attempt to extensively correct segmentation errors, thus, massively increases the search space for correcting. If, however, tokenisation was not carried out solely on the basis of how a token is delimited, but the internal structure of a token was taken into account instead, one might expect a tokenisation scheme that is less sensitive to segmentation errors produced by the recognition system. Example (b) shows how a more fine-grained tokenisation can simplify

the correction procedure, in that the search space is reduced. The resulting segments do not necessarily correspond to any linguistic categories; they might, however, help localise and fix segmentation errors. In the example given, all errors can be addressed within the scope of a single token, no $1{:}n$ correspondences have to be considered.

2 Related Work

2.1 Automated OCR Error Correction

The field of automated OCR error correction has gained attention over the last years, keeping up with a growing number of large-scale digitisation projects.[1] Most of the work is concerned with cleaning up the output of an off-the-shelf OCR system, i. e. post-processing, although there are attempts at tweaking the performance of the system itself, such as Heliński et al. (2012). While purely dictionary-based correction attempts for OCR errors have not been very popular lately, there is still, work on efficient dictionary lookup, designed for the use with OCRed search terms and similar scenarios (Mihov and Schulz, 2004; Mihov et al., 2007). Lund and Ringger (2009) and Volk et al. (2010) achieve text improvements by combining the output of multiple OCR systems, following the idea that different recognition techniques lead to different errors.

Corpus-based post-correction of OCR errors has found more and more proponents in the past, thus according with a general trend in many NLP areas. Among other advantages, corpus-based correction – as opposed to a dictionary-driven approach – enables exploitation of context information, which allows for addressing *real-word errors*, i. e. OCR misrecognitions that result in another existing word (e. g. *Negierung* 'negation' instead of *Regierung* 'government'), as opposed to *non-word errors*, which are misspelt words (e. g. *Rcgiernng*). As an early example, Tong and Evans (1996) investigate the use of a bigram language model for correcting OCR errors. The corrections are performed on word level, using a lexicon derived from a training set of error-free texts. The authors tackle both non-word and real-word errors and report error reduction rates up to 60 % for plain alphabetic tokens. Bassil and Alwani (2012) perform corpus-based corrections with the Google Web 1T 5-Gram Data Set. Their approach performs very well on noisy OCRed text, although the small size of the test set (less than 300 running tokens) gives it only limited evidence. More of a bootstrapping approach is followed by Reynaert (2008). He pursues the idea that OCR errors are unsystematic noise that can be filtered out without the use of clean text.

Some work also addresses segmentation errors. For example, Reynaert (2004) builds a corpus-derived lexicon containing word bigrams, wich enables a chance of correcting run-on tokens. Interestingly, he later states that he "do[es] not here attempt to resolve run-ons" (Reynaert, 2006, p. 90). Kolak et al. (2003) explicitly tackle both kinds of segmentation errors by allowing splits in the lexicon words and the OCR tokens.

2.2 Unsupervised Text Segmentation

Most NLP tasks operate at the level of words. The task of splitting text into words needs some attention in the case of continuous sequences, as with speech recognition or in the case of orthographies lacking word boundaries such as Chinese, see e. g. Chung and Gildea (2009), or when dealing with phonemical transcriptions (Goldwater et al., 2006). It is usually referred to as *word segmentation*. In contrast to this, *tokenisation* is the task of achieving the same goal for texts containing word dividers (mostly blank spaces). Although tokenisation seems to be quite straightforward a task, there are still innovations in the field, like the proposal by Barrett and Weber-Jahnke (2011), who aim at performing tokenisation and part-of-speech tagging simultaneously.

Tokenisation may also be difficult in the case of untrusted input. Wrenn et al. (2007) attempt the segmentation of texts produced by clinicians, which have a high spelling-error rate (including segmentation errors), causing troubles to a standard tokeniser. The authors introduce word boundaries using a technique borrowed from unsupervised morphological segmentation,[2] which

[1] See e. g. Holley (2009) for a large Australian newspaper digitisation program, Jisc (`http://www.jisc.ac.uk/`) for a list of ongoing projects on the British Isles, or the IMPACT initiative (Tumulla, 2008).

[2] See the comprehensive work by Hammarström and Borin (2011) for an overview of the field.

	1	1	2	1	1	2	20	5	13	25		←
	d	i	s	t	u	r	b	a	n	c	e	
→		15	24	24	8	2	2	4	2	1	1	

Figure 2: LPV (first row) and LSV (last) counts for the word *disturbance*, based on a list of dictionary entries (Harris, 1967, p. 69). The general tendency of decreasing numbers is interrupted by "peaks" at positions with higher variability, e. g. between *disturb* and *ance* both in left-to-right and right-to-left reading.

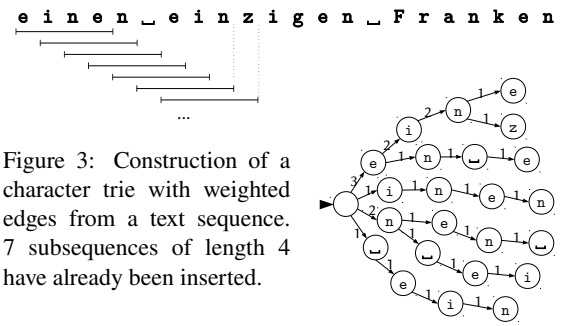

Figure 3: Construction of a character trie with weighted edges from a text sequence. 7 subsequences of length 4 have already been inserted.

was originally applied to single words and short utterances. Wrenn et al. adapt the method to work with running text of arbitrary length.

Golcher (2006) proposes a comprehensive approach at segmenting text in an unsupervised manner, addressing text segmentation, morphological decomposition, multiword unit detection, and compound analysis at the same time. He uses a combination of different statistical measures to segment continuous text into useful units. Unfortunately, the author did not perform an extrinsic evaluation, such as applying the segmented text to an information retrieval or machine translation system, by which means the advantages of the proposed segmentation could have been shown.

3 Methods

In a series of experiments in this and earlier work (Furrer, 2013), I examined the use of an alternative text segmentation scheme, as opposed to traditional tokenisation. The effects of the segmentation are measured by the performance of an automated error correction system.

3.1 Text Segmentation

Before being processed by the correction module, all text needs to be segmented into basic units. The text segmentation method presented here induces segment boundaries from the distribution of characters. It is based on the work of Wrenn et al. (2007) and goes back to the *letter successor variety* method (LSV), which was introduced by Harris (1955) and given its name by Hafer and Weiss (1974). The intuition behind LSV is that morpheme boundaries can be inferred from statistical properties of the characters found in a list of words or short phrases. For a set of words that share a common prefix x, LSV(x) is defined as the number of distinct characters that succeed x in these words. In figure 2, the bottom row lists LSV counts for every character transition in a test word, based on counts for all entries of an English dictionary. For example, if a word begins with the letter *d*, the second letter can be chosen from 15 possibilities, one of which is *i*. In all words beginning with *distu*, the sixth position is occupied by one of 2 distinct letters. Analogously, LSV can be counted backwards, i. e. counting the distinct letters preceding a shared suffix (see the top row in figure 2). This is called *letter predecessor variety* (LPV). In order to manage continuous text of arbitrary length, the context for computing LSV/LPV is limited to a *window* of k characters (Markov assumption). The windowed subsequences are stored in a character trie as is illustrated in figure 3.

During segmentation, the entire text collection is read twice: In a learning step, the character distribution is gathered from all texts and stored in two character tries – one for the forward, and one for the backward reading. Subsequently, this global information is used to find good split points locally. Given an input sequence s, a pair of character tries, and a minimal peak threshold, the segmentation routine splits s *exhaustively* into adjacent subsequences. Whitespace characters are thus not removed, but retained in the segments.

For every character transition i in s, a fragility score f is computed on the basis of LSV. The algorithm aims at finding peaks in the sequence of LSV values, i. e. values that are greater than their immediate neighbours. If there is a peak at transition i, then the LSV drop to both its neighbours is summed:

$$f_s(i) = \begin{cases} \Delta_s^<(i) + \Delta_s^>(i) \\ \quad \text{if } \Delta_s^<(i) > 0 \wedge \Delta_s^>(i) > 0 \\ 0 \quad \text{otherwise} \end{cases} \qquad (1)$$

where $\Delta^<$ and $\Delta^>$ are the increasing and decreasing side of the peak, respectively. For a character window of length k, the definitions of $\Delta^<$ and $\Delta^>$

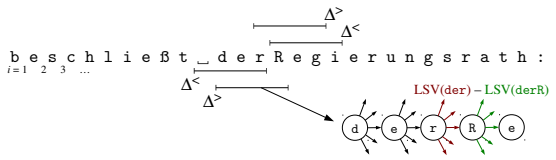

Figure 4: Illustration of a segmentation example.

are narrowed down to the following:

$$\begin{aligned}\Delta^{<}_{s,k}(i) &= \mathrm{LSV}(s^{i}_{i-k+2}) - \mathrm{LSV}(s^{i-1}_{i-k+2}) \\ \Delta^{>}_{s,k}(i) &= \mathrm{LSV}(s^{i}_{i-k+3}) - \mathrm{LSV}(s^{i+1}_{i-k+3})\end{aligned} \tag{2}$$

By looking separately at each side of i, $\Delta^{<}$ and $\Delta^{>}$ capture a maximum of context each. For example, with a window size $k = 5$ and $s =$ `beschließt␣derRegierungsrath`:, $\Delta^{<}_{s}(i = 14)$ is calculated using the following subsequences (see also figure 4):

$$\begin{aligned}\Delta^{<}_{s,5}(14) &= \mathrm{LSV}(\texttt{␣der}) - \mathrm{LSV}(\texttt{␣de}) \\ \Delta^{>}_{s,5}(14) &= \mathrm{LSV}(\texttt{der}) - \mathrm{LSV}(\texttt{derR})\end{aligned} \tag{3}$$

Analogously, the fragility score is computed for the backward reading of s. If the summed scores at transition i reach or exceed the peak threshold, a segment boundary is inserted. Figure 5 shows an example of segmented text.

3.2 Error Correction

I modelled the automated error correction framework closely after the Hidden Markov Model (HMM) proposed by Tong and Evans (1996), which is theoretically well founded and easily adaptable to new data. By realising the digitisation chain as a Markov model, one assumes that the original text can be cleaned from the transmission noise by means of statistical properties.

In the experiments by Tong and Evans, these properties are estimated from a collection of clean texts. The observations are the output produced by the OCR system, cut into processable units (henceforth *segments*) in the preceding segmentation routine. The hidden states are the correct segments that have to be predicted by the correction system. The correction is performed by finding the best path through all possible hidden states, which is done with the Viterbi algorithm.

An important thing to note is that the distinction between error *detection* and error *correction* is absent from this approach. In a HMM, correcting a text segment means predicting a hidden label that looks different from the observed token; not correcting – which is the most frequent operation – means predicting a label that happens to equal the observation. This behaviour is controlled by the emission probability.

`Hier ward mir zum ersten Male die Möglichkeit klar,`
`den Koloss von Südwesten zu bezwingen. Mit dem Fernrohr`
`untersuchte ich die östlichen Wände des Aletschhorns`

Figure 5: Example of segmented text. Segment boundaries are indicated by change of shading. The character distribution properties were learnt with a window size of 5 and the segmentations were carried out with a peak threshold of 10.

The best sequence of correct segments W^{best} is determined by the conditional probability $P(W|O)$. Using Bayes' theorem, this is equivalent to:

$$W^{\text{best}} = \arg\max_{W}(P(W) \times P(O|W)) \tag{4}$$

$P(W)$ and $P(O|W)$ are the products of the transition and emission probabilities, respectively, for a sequence of observations O and all possible correction sequences W. $P(W)$ is estimated with a statistical language model using the tool `KenLM` (Heafield, 2011), which implements *modified Kneser-Ney* smoothing (Kneser and Ney, 1995; Chen and Goodman, 1998). The language model was trained on a normalised version of the text collection.[3] $P(O|W)$ is estimated from the frequencies of observed human corrections by maximum likelihood estmation (MLE). It expresses the distribution of noise that has been introduced by the OCR process. In order to reduce the search space for potential corrections, I used `simstring` (Okazaki and Tsujii, 2010) for fast retrieval of candidate segments with a minimal cosine similarity of 0.6.

The probability of an OCRed segment o being produced from a true segment w is based on the *minimum edit distance* between the two segments. This means that the cost of transforming w into o is described in terms of edit operations, i. e. inserting, deleting, or substituting a character. By finding the minimal set of edit operations, the characters of w and o are aligned. Based on the confusion probability of each character alignment pair, $P(o|w)$ can be estimated:

$$P(o|w) = \prod_{i,j \in \mathcal{A}_{w,o}} P_{\text{conf}}(o_i|w_j) \tag{5}$$

[3] All letters were converted to lower-case and sequences of digits were replaced by '0'. Please note that this normalisation is only applied when estimating and querying the transition probability, but not for the emission probability.

where $o_1, \ldots, o_n$ and $w_1, \ldots, w_m$ are the characters in o and w, respectively, $\mathcal{A} \subseteq \{0, \ldots, n\} \times \{0, \ldots, m\}$ is the set of alignments between o and w, and $P_{\text{conf}}(x|y)$ is the confusion probability of the aligned characters x and y, estimated with MLE. In order to account for unseen character pairs, the estimations are smoothed with a discount factor α in the range $[0, 1]$. If there is no empirical estimation for a character pair o_i, w_j, the following back-off model is applied:

$$P_{\text{backoff}}(o_i|w_j) = \begin{cases} \alpha & \text{if } o_i = w_j \\ \frac{1-\alpha}{|A|-1} & \text{otherwise} \end{cases} \quad (6)$$

where $|A|$ is the size of the training data alphabet.

4 Evaluation

In the present work, the outcome of "unorthodox" text segmentation is investigated by measuring its impact on a subsequent NLP application, namely the automatic correction of OCR errors. In an extrinsic evaluation like this, the usefulness of the intermediate step – the segmented text – is only determined by the improvements of the complete system. It does not matter if the resulting segments do or do not agree with human annotations, correlate with usual tokenisation, or correspond to linguistically approved entities.

For the present experiments, I worked with a collection of alpine texts from the 19th century. The text collection consists of 35 volumes of the yearly publications of the Swiss Alpine Club. The books were digitised using OCR in the *Text+Berg digital* project,[4] located at the University of Zurich. Most of the texts are written in German (approximately 90 % of the sentences) and French (10 %), with some portions in English and Italian (0.1 % each). They were published between 1864 and 1899 and sum up to a total of more than 21 000 pages with approximately 560 000 word tokens. The text quality in terms of OCR accuracy is acceptable, but far from perfect. The OCR process was challenged by many factors, such as historical spelling, a high rate of special vocabulary (place names, scientific terms, regional language variations), mixed paper quality, and complex layout (tables, equations).

The text versions used in this work are taken from an offspring project called *SAC-Kokos*,[5] which runs an online platform for publishing and improving the OCRed texts. The project is built on the idea of crowd-sourcing: users may read the texts online and edit them in an easy click-and-type manner whenever an error is encountered. Over the last months, the text quality has been considerably improved by enthusiasts who read through the texts and correct OCR errors.

Throughout the experiments, I used two snapshots of the text collection: one taken at an early stage of the project, when the texts still were close to the raw output of the OCR process (henceforth "*oldest*"), and a very recent one, reflecting many improvements by human editing ("*newest*"). These two versions of the same text collection are used to measure how well the correction system is able to imitate human text correction.

4.1 Evaluation Setup

For creating a test set, I collected a subset of pages with a minimal length of 100 words. I further excluded pages with a character edit ratio below 0.2 % or above 2 %, as it is very likely that pages with a very low change rate have not been thoroughly reviewed yet, while pages with too many edits probably reflect problems outside the reach of an automated correction attempt, such as rearrangements of longer text parts. From this subset, which comprises about 60 % of all pages, I sampled 1000 pages as a test set, and another 1000 pages as a tuning set. The remainder of the (unfiltered) collection was used for training.

The evaluation setup is modelled towards a realistic scenario, where a collection of noisy texts is available, but only a limited amount of clean data. Thus, I used the *oldest* version of the data for training the transition probabilities of the correction HMM. The emission probabilities were estimated from the differences in the *newest* and *oldest* versions of the tuning set pages.

Both training and test data were segmented with the described method. Based on experience with prior experiments, I set the character-trie window and the peak threshold to values of 5 and 10, respectively. Additionally, I ran a separate instance of this experiment with tokenised data, created with a simple regular-expression based tokeniser. For each of the two instances, I carried out multiple runs by varying the value of the discounting parameter α, which affects the emission probabil-

[4] http://textberg.ch/

[5] http://kokos.cl.uzh.ch/

	segmented		tokenised	
	$\alpha = .97$	$\alpha = .9$	$\alpha = .97$	$\alpha = .9$
Δ_E	-16	-12	-11	-12
mod.	30	35	13	15
TP	23	23.5	12	13.5
FP	7	11.5	1	1.5
Prec.	76.67 %	67.14 %	92.31 %	90.00 %
Rec.	0.21 %	0.21 %	0.11 %	0.12 %

Table 1: Error reduction for segmented and tokenised data.

ity of the HMM.

After estimating the HMM weights from the training data, the correction system processed the *oldest* version of the test data. Using the *ISRI OCR-Evaluation Frontiers Toolkit* (Rice, 1996), I measured the OCR quality before and after applying the corrections, by assuming that the *newest* test data version is a reasonable approximation of ground truth data.

4.2 Results

Evaluation results are given in table 1. Each column represents a different experimental configuration. The first row (Δ_E) shows the respective error reduction rate (in characters). The total number of modifications made by the system is given in the second row (number of modified segments/tokens, but usually only one character is affected). The following rows give figures for true and false positives (TP, FP) as well as precision and recall. A modification by the system was counted as TP if it reduced the error rate, but as FP if it performed an over-correction (i. e. introduced a new error, at least from the perspective of the *newest* data). In some cases, the system spotted an error correctly, but failed at correcting it (e. g. when deleting a misrecognised character rather than replacing it), which is a neutral modification with respect to the error rate; I counted these cases as half TP, half FP. The recall figures are based on the total number of character errors in the test set, which is 11 100.

5 Discussion

All systems show a moderate error reduction. However, the 60.2 % error reduction reported by Tong and Evans (1996, p. 96) cannot be reproduced, for the systems are very conservative (one would like to say "shy"). The reason for the low recall seems not to lie in the non-standard segmentation, since the tokenisation-based systems are just as cautious. Rather, it seems that the correction model does not adapt well to the present-day requirements of error correction. In fact, one of the key differences between the same task in the 1990s and in the 2010s is the initial accuracy of the noisy documents: While the overall word-error rate in Tong and Evans' texts is 22.9 %, it is as low as 1.7 % in my test data.[6] This means that the task of finding these well-hidden errors is now harder by an order of magnitude.

Comparing the results of the different systems, the segmentation-based approach generates more TP and leads to a greater error reduction, while tokenisation yields a higher precision. The effect of the discounting factor α is small. It seems to make the systems slightly more audacious when reduced (which gives more probability mass to unseen character substitutions); this comes at a cost in precision, however.

A qualitative analysis of the data by inspecting the performed modifications exemplifies the advantages of the unsupervised segmentation approach. Besides generally good corrections like *Glär-nisch* → *Glärnisch* (a mountain) or *HUtten* → *Hütten* ('huts'), which could also be detected in a word-based correction attempt, there are cases that clearly profit from the non-standard segmentation. The French phrase *II y a* → *Il y a* ('there is') and the spaced abbreviation *8. A. C.* → *S. A. C.* were each treated as a single segment and could be safely corrected, while the single tokens *II* and *8.* are not that easily identified as errors. Furthermore, *tobeis* → *tobels* corrects the last part of the compound place name *Welschtobel* (in genitive case), which is presumably supported by occurrences of this segment in different compounds.

While the overall performance of the presented error reduction system is not overwhelming, the unsupervised segmentation scheme is able to address certain kinds of errors that are harder to find by word-based correction systems. In future work, I intend to test the unsupervised segmentation with a more sophisticated correction algorithm.

[6] Measured by the *newest* data, which might be too low an estimate, since there might still be uncorrected errors.

References

Neil Barrett and Jens H. Weber-Jahnke. 2011. Building a biomedical tokenizer using the token lattice design pattern and the adapted Viterbi algorithm. *BMC Bioinformatics*, 12(S-3).

Youssef Bassil and Mohammad Alwani. 2012. OCR context-sensitive error correction based on Google Web 1T 5-Gram data set. *American Journal of Scientific Research*, 50(50), February.

Stanley F. Chen and Joshua T. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report tr-10-98, Harvard University.

Tagyoung Chung and Daniel Gildea. 2009. Unsupervised tokenization for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, EMNLP '09, page 718–726. ACL.

Lenz Furrer. 2013. Unsupervised text segmentation for correcting OCR errors. Master's thesis, University of Zurich.

Felix Golcher. 2006. Statistical text segmentation with partial structure analysis. In *Proceedings of KONVENS*, page 44–51.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, ACL '06, page 673–680.

Margaret A. Hafer and Stephen F. Weiss. 1974. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10(11-12):371–385.

Harald Hammarström and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350.

Zellig S. Harris. 1955. From phoneme to morpheme. *Language*, 31(2):190–222.

Zellig S. Harris. 1967. Morpheme boundaries within words: Report on a computer test. In *Transformations and Discourse Analysis Papers*, page 68–77. Department of Linguistics, University of Pennsylvania.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Edinburgh, UK, July. Association for Computational Linguistics.

Marcin Heliński, Miłosz Kmieciak, and Tomasz Parkoła. 2012. Report on the comparison of Tesseract and ABBYY FineReader OCR engines. Technical report, Poznań Supercomputing and Networking Center (PCSS), Poznań.

Rose Holley. 2009. How good can it get? analysing and improving OCR accuracy in large scale historic newspaper digitisation programs. *D-Lib Magazine*, 15(3/4).

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume I, page 181–184, Detroit, Michigan, May.

Okan Kolak, William Byrne, and Philip Resnik. 2003. A generative probabilistic OCR model for NLP applications. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 1 of *NAACL '03*, page 55–62, Stroudsburg, PA, USA. Association for Computational Linguistics.

William B. Lund and Eric K. Ringger. 2009. Improving optical character recognition through efficient multiple system alignment. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '09, page 231–240, New York, NY, USA.

Stoyan Mihov and Klaus U. Schulz. 2004. Fast approximate search in large dictionaries. *Computational Linguistics*, 30:451–477.

Stoyan Mihov, Petar Mitankin, Annette Gotscharek, Ulrich Reffle, Klaus U. Schulz, and Christoph Ringlstetter. 2007. Tuning the selection of correction candidates for garbled tokens using error dictionaries. In *Finite State Techniques and Approximate Search: Proceedings of the First Workshop on Finite-State Techniques and Approximate Search*, page 25–30, Borovets, Bulgaria.

Naoaki Okazaki and Jun'ichi Tsujii. 2010. Simple and efficient algorithm for approximate dictionary matching. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, page 851–859, Beijing, China, August.

Martin Reynaert. 2004. Text induced spelling correction. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, page 834–840, Geneva, Switzerland, Aug 23–Aug 27. ICCL.

Martin Reynaert. 2006. Corpus-induced corpus cleanup. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, LREC 2006, page 87–92. European Language Resources Association (ELRA).

Martin Reynaert. 2008. Non-interactive OCR post-correction for giga-scale digitization projects. In *Proceedings of the 9th International Conference on Computational Linguistics and Intelligent text processing*, CICLing'08, page 617–630, Berlin, Heidelberg. Springer-Verlag.

Stephen V. Rice. 1996. *Measuring the Accuracy of Page-Reading Systems*. Ph.D. thesis, University of Nevada, Las Vegas.

Xian Tong and David A. Evans. 1996. A statistical approach to automatic OCR error correction in context. In *Proceedings of the Fourth Workshop on Very Large Corpora*, WVLC-4, page 88–100.

Martina Tumulla. 2008. IMPACT: Improving access to text. *Dialog mit Bibliotheken*, 20(2):39–41, July. (German article).

Martin Volk, Torsten Marek, and Rico Sennrich. 2010. Reducing OCR errors by combining two OCR systems. In *ECAI 2010 Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, LaTeCH 2010, page 61–65, August.

Jesse O. Wrenn, Peter D. Stetson, and Stephen B. Johnson. 2007. An unsupervised machine learning approach to segmentation of clinician-entered free text. In *Proceedings of the AMIA 2007 Annual Symposium*, page 811–815.