# Low-resource Post Processing of Noisy OCR Output for Historical Corpus Digitisation

**Caitlin Richter, Matthew Wickes, Deniz Beser, Mitch Marcus**
University of Pennsylvania
Departments of Linguistics and Computer and Information Science, 3330 Walnut Street, Philadelphia PA 19104
{ricca, wickesm, dbeser}@seas.upenn.edu, mitch@cis.upenn.edu

**Abstract**

We develop a post-processing system to efficiently correct errors from noisy optical character recognition (OCR) in a 2.7 million word Faroese corpus. 7.6% of the words in the original OCR text contain an error; fully manual correction would take thousands of hours due to the size of the corpus. Instead, our post-processing method applied to the Faroese corpus is projected to reduce the word error rate to 1.3% with around 65 hours of human annotator work. The foundation for generating corrected text is an HMM that learns patterns of OCR error and decodes noisy OCR character sequences into hypothesised correct language. A dictionary augments the HMM by contributing additional language knowledge, and a human annotator provides judgements in a small subset of cases that are identified as otherwise most prone to inaccurate output. An interactive workstation facilitates quick and accurate input for annotation. The entire toolkit is written in Python and is being made available for use in other low-resource languages where standard OCR technology falls short of desirable text quality. Supplementary analyses explore the impact of variable language resource availability and annotator time limitations on the end quality achievable with our toolkit.

**Keywords:** Faroese, OCR, low resource corpus development, human-assisted post processing

## 1. Introduction

### 1.1. Motivation

We provide a toolkit to improve unsatisfactory text quality in a newly digitised Faroese corpus (Føroyamálsdeildin, nd). The 2.7 million word corpus contains historical interviews of unique linguistic and cultural value, which would provide valuable material for research in a variety of disciplines if it were readily accessible (e.g. Heycock and Wallenberg, 2013; Galbraith, 2016). The interviews were recorded on tape up to 50 years ago, transcribed by typewriter, and later scanned (Figure 1) before being input to unknown commercial optical character recognition (OCR) software for text extraction.

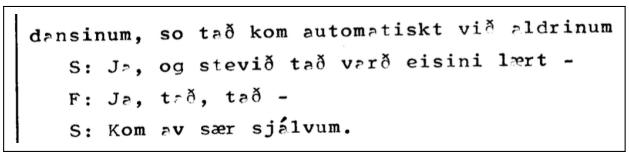
dansinum, so tað kom automatiskt við aldrinum
S: Ja, og stevið tað varð eisini lært -
F: Ja, tað, tað -
S: Kom av sær sjálvum.

Figure 1: Partial scanned typewriter page of Faroese interview transcription.

This process significantly corrupted the text with incorrect recognition. A lack of enough language-specific data for good Faroese language models in the initial OCR process had a large role in damaging the text quality; many of the errors produce tokens with distinctly un-Faroese letter sequences, or even numbers and punctuation in the middle of words (like OCR error form *r!ey^ur* for *deyður*). Close to 8% of tokens (words) contain at least one error, suggesting a rough expectation for on average every sentence to contain one incorrectly recognised (non-)word. This error rate severely degrades the usability of the corpus for NLP, computational linguistic research, or any interest that relies on basic text search to identify relevant documents within the 2.7 million word corpus.

Faroese is spoken by only about 50,000 people on the Faroe Islands, located between Norway, Scotland, and Iceland (Thráinsson et al., 2012). It is a Germanic language whose nearest relative is Icelandic; the two are not normally mutually intelligible in speech to unpractised listeners, and Faroese is considerably more different (certainly not mutually intelligible) from its next closest relatives, the larger and better-resourced Mainland Scandinavian languages Danish, Norwegian, and Swedish. However, many native speakers of Faroese do speak Danish as a second language, in current times often used for university education or employment in Denmark (while Faroese is the language of home, daily life, school, etc. on the Faroe Islands). This reflects past Danish rule of the Faroe Islands; of particular note to present purposes, this rule included a period of around 300 years in which the Faroese language was generally banned from official and written contexts (Thráinsson et al., 2012). The modern orthography used to transcribe the corpus we correct was developed in the late 1800s, after the language ban was lifted, by linguist V.U. Hammershaimb; it incorporates etymological reconstruction and morphophonemic (rather than phonetic) representation, which has the effect that it is not inherently biased to resembling the pronunciation of particular dialects. In this corpus, while lexical items will vary according to speakers' dialects, the spelling itself can be considered a stable standardised system. The Faroese alphabet officially has 29 letters, but in practise a few more are found - variant conventions like **ø** and **ö** which are intended as the same letter in this context, and foreign borrowings like **å** and **c**. We suspect, based on certain qualitative patterns in recognition, that the OCR system that produced the Faroese corpus text may have had Danish as its primary language, with the Faroese alphabet traded in.

### 1.2. Approach

We developed a suite of post-processing tools that a corpus can be fed through, obtaining adequate machine-readable text despite the excessive error rate from standard OCR. The toolkit is general and we are distributing it for use in any corpora where conventional OCR lacking sufficient language-specific resources is inadequate.

Some human annotation seems necessary in the digitisation of the Faroese corpus. In the first place, automated OCR is unsatisfactory; then, we find also that building an entirely automated post-processing system, although it improves the text, still leaves far too much error. However, incorporating a moderate investment of annotator time during post-processing leaves an acceptable token error rate below 1.5% in this low-resource setting.

Automated tools handle most of the corpus without human intervention, but also identify scenarios where automated accuracy tends to be lowest. In these difficult cases, the automatic tools restrict the problem to a forced choice of at most 3 potential corrections for an error, which are then sent to a human annotator for adjudication. Our projections estimate a final token error rate of 1.3% with around 65 hours of annotator work to process the entire Faroese corpus.

The only alternative in order to produce machine-readable text of acceptable quality would be manual human work. A human editor moving quickly can correct around 1,000 words of text per hour, for an expectation of 2,700 hours of work to process the Faroese corpus while still being prone to some degree of mistakes and inattention leading to a comparable token error rate in the corpus. As an alternative to correction, the entire corpus could be transcribed from scans by typists, skipping the OCR entirely; this too would take substantial time for 2.7 million words, and the cost of hiring professional transcribers is often prohibitive for academic or community owners of language resources being digitised. Our toolkit is designed to achieve similar final corpus quality at much lower annotator cost.

### 1.3. Contributions

The Faroese corpus whose noisy OCR we correct represents a general problem in the dissemination of physical documents in low resource language varieties. A basic pipeline is that a physical document is scanned to a digital image, the digital image has machine-readable text extracted by OCR, and then the text is available for distribution. OCR is considered a solved problem for modern type in large-population languages, but standard technologies perform poorly in low resource language varieties, or when source documents are physically degraded by time, environment, or show visual variation due to the original printing method (Afli et al., 2016). Digitised resources could be distributed as scanned images rather than machine readable text, and communities may still access the heritage and culture represented in the resources. However, text facilitates better distribution due to reduced file size, and large corpora are not easily navigable for any purpose without searchable text. Text is of course also essential for NLP, computational linguistics, and other research uses of corpora. Therefore, we consider machine-readable text a highly important element for any digitised text corpus. Our toolkit becomes a post-processing stage in the resource dissemination pipeline, improving error-filled OCR output before the text is distributed. The toolkit's code, in Python, is being made available for general use.

## 2. Overview of Toolkit

A suite of five components and their relations are introduced briefly here, before individual presentation in §3.

**Character decoding.** The core generation of OCR corrections uses a character HMM to decode noisy input text into $k$-best candidates for correct Faroese.

**Alignment.** Word-aligned parallel texts match noisy OCR with its gold standard correction. Aligned texts are used to train the character HMM and for development/evaluation.

**Dictionary.** A dictionary helps filter out spurious (non-word) character sequences generated by HMM decoding.

**Heuristic Decisions.** A central decision module draws on the output of character decoding and the language dictionary to select output forms or determine when to seek human adjudication between $k$-best candidates.

**Interface.** The human annotator is presented with a potential OCR error and up to three candidate corrections in an interactive workstation, and inputs their selected correction.

## 3. System Description

### 3.1. Text Alignment

Aligned texts (Table 1) contain word pairs used to train models of OCR character error. Aligned parallel text like this is also required for system tuning and evaluation.

| Correct | Original |
|---|---|
| tøðini | tøðini |
| v**ó**rðu | v**ú**rðu |
| borin | borin |
| út | út |
| **á** | **§** |
| bøin | bøin |
| **í** | **i** |

Table 1: Sample of aligned text, including OCR errors (underlined).

Corrected versions of several Faroese corpus texts were available as they had been previously produced and used for linguistics research. Aligning a noisy OCR document to its corrected version is not trivial, because the two texts normally have different token counts, different word and sentence segmentations, and in some cases different line divisions. However, both should contain corresponding elements in the same order. We used the Needleman-Wunsch sequence alignment algorithm (familiar also as 'dynamic time warping' when applied to speech) on pairs of noisy and corrected Faroese text files to find optimal alignments (Kleinberg and Tardos, 2006).This dynamic programming algorithm creates a grid of character (mis)matches between two files, and uses these to iteratively calculate the best word-level alignment minimising overall mismatch. A fixed window for mismatch length reduces runtime.

### 3.2. Character Decoding

To generate corrections for tokens in noisy OCR text, we use a hidden Markov model to decode the original OCR into hypothesised correct Faroese output. The decoding process takes a token from the input file and identifies the sequence of actual Faroese characters most likely to have been recognised by OCR as that token, which may or may not be identical to the original character sequence. We use a modified version of the Viterbi algorithm to find the top-$k$ most probable character sequences, instead of the standard single best character sequence. This modified algorithm uses beam search, maintaining a list of the $k$-best most probable sequence possibilities at each time slice within the decoding process.

The states of the HMM are the characters present in the corrected training data: the Faroese alphabet, numbers, and punctuation. The HMM output symbols are the slightly larger set of characters present in the original noisy OCR, such as đ alongside Faroese ð. Emission parameters of the HMM are learned from parallel training data, while initial and transition probabilities are learned from corrected texts. Laplace smoothing enables generalisation to character transitions and OCR errors not seen in the training data.

| | |
|---|---|
| Initial Error | 7.8% |
| Final Error | **5.4%** |
| Introduced Error | 0.8% |
| Tokens Successfully Corrected | 3.2% |

Table 2: Operation of HMM alone on dev set. All quantities are expressed by percentages of corpus tokens (words).

The HMM is trained on 170k words of parallel Faroese data, and decoding uses beam width $k$=4. This HMM alone is not able to effectively correct OCR errors. Table 2 illustrates projected performance for always selecting the most probable (Viterbi) decoding on a test set of 70k words held out from its training data. This poor performance motivates our strategy of overgeneration ($k$-best decoding) before taking additional steps to refine the OCR correction.

The simple HMM topology that we use is capable of correcting single character errors, but the noisy OCR on the Faroese corpus also produced errors of multiple characters read as a single character and single characters read as multiple characters. These errors are generally rare, but two types occur with higher frequency: 'll' read as 'H' and 'rn' read as 'm'. These two error types are present in 0.18% of the tokens in the parallel Faroese data.

Correction of these errors is done after computing the $k$-best decodings of the original token, and is corpus-specific. If none of the $k$-best decodings are present in the dictionary then the token is checked for whether it satisfies the conditions for any of the character mis-segmentation errors known to be frequent in the corpus.

1. If the original token contains an 'H' in a non-initial position, then any occurrences of 'H' are replaced by 'll' and the $k$-best decodings are then found for the new token. The $k$-best among the original decodings and the new decodings are then selected. Only non-initial 'H' is replaced because no words beginning with 'll' are present in the Faroese dictionary.

2. If the original token contains an 'm' then variant tokens are created substituting 'rn' in place of 'm'. Every occurrence of 'm' is either replaced by 'rn' or left unchanged. The $k$-best decodings are computed for each of these variants and then the $k$-best among those and the original decodings are selected.

If a token satisfies conditions for both errors then both processes are applied.

The two strategies applied here, complete replacement in (1) and variable mixing in (2), can be generalised to other OCR errors of this nature in corpora where such errors are more prevalent.

### 3.3. Dictionary

The Faroese dictionary contains around 450k types. It was created originally for spell checking (J. S. Andersen, 2001), and lightly augmented for our corpus (for example, common abbreviations like *Fx* 'speaker' and *Fxx* 'speaker 2'). Around 5% of tokens in the corrected text are still not present in the dictionary.

This dictionary acts as a filter on the $k$-best decodings, when at least one candidate is in the dictionary and at least one is not. In these cases, a candidate that is in the dictionary may be a better choice than one that is not, even if this is counter to their decoding probabilities. For example, the 4-best HMM decodings for the token *stðani* are [stóani stðani stúani síðani]; of these, the fourth *síðani* is correct and is present in the dictionary, while none of the other three generated forms are members of the dictionary. Similarly, the two best decodings for *slektaðu:* with the token ending in a colon are [slektaðu: slektaður]. Here *slektaður* is correct, and it is in the dictionary while *slektaðu* (punctuation stripped for dictionary check) is not. Therefore, if some but not all generated candidates are found in the dictionary, non-members can be removed from the candidate set. The dictionary is also used to assess whether to replace original tokens with any non-identical candidate at all: if the original token is already in the dictionary, then it is less likely to be an error in need of correction.

### 3.4. Central Decisions

OCR correction is ultimately managed by a module that combines all available information with heuristic handling strategies, and either selects the best word decoding or solicits human judgement.

The information available to the central decision module includes an original token, four candidate decodings of it (one of which may be identical to the original), probabilities for each decoding under the HMM, and binary dictionary membership checking. Decisions about which form to use as output, or whether to ask a human annotator to adjudicate between a set of candidates, must derive from this information.

The current decision procedure is in broad summary:

1. Construct an initial ranked candidate list, from the original token and *k*-best list: [ORIG, K1, K2, K3, K4]

2. Check basic attributes of list members, such as whether they are members of the dictionary and whether they are identical with the original input form.

3. Based on token and candidate attributes, handle the correction by either making no change, changing to the highest-ranked non-identical candidate present in the dictionary, or seeking human annotator judgement.

The decision procedure applies invariantly whether the best decoded candidate (K1) is different or identical to the input, that is, whether the character model estimates that the original OCR contains an error or not.

This basic outline is implemented through a set of heuristics that allow different classes of tokens to be handled differently, increasing correction accuracy and maximising efficiency for the annotator by better targeting their efforts. Tokens that are being corrected can be classified based on four binary attributes visible to the system at correction time: whether the top-ranked generated candidate K1 is identical to the input token; whether the input token is in the dictionary; whether the best generated candidate K1 is in the dictionary; and whether *any* generated candidates are in the dictionary. Due to frequent logical dependencies between the attribute values, 9 classes are created. Each class can receive an independent decision for how to best handle its tokens, with these heuristic decisions optimised on a dev set: possible choices for each are to select the original token, the top generated candidate, the top candidate appearing in the dictionary (if present), or human annotation. The decision for how to handle each class is determined by oracle evaluation run on a dev set of parallel original/corrected (gold standard) text. This shows what the optimal decision on the dev set would have been, as well as what percentage of the corpus falls into each of the nine classes so that annotator time can be allocated to make the largest possible impact while minimising workload. If the dev set reasonably represents the corpus as a whole, then the optimal heuristic settings for the dev set should lead to accurate and efficient corrections when used to correct the entire corpus.

#### 3.4.1. Fit to Faroese corpus

We used four-fold cross validation to set heuristic options (based on 10,000 words parallel text) and test the projected performance of these settings when applied to other text (another 10,000 words). The parallel text used for evaluation is 500-word excerpts of 40 texts from the corpus, randomly sampled with restriction of no more than one text per interviewee to enforce wide coverage of vocabulary, year interviewed/transcribed, and individual typewriters. The corrected versions of the parallel texts were finalised with reference to scans of their original typewritten copies to produce a gold standard for precise evaluation.

In four-fold cross validation, there was no variation in which heuristic settings were found to be optimal for different random 10,000-word segments of the dev data, so we treat this as a single correction system. The settings found for Faroese correction can be summarised across the 9 cases in the following decision procedure:

1. If the top-ranked generated candidate K1 is in the dictionary, use K1 (which may be identical to the input).

2. Else, if the original input is in the dictionary, use that input.

3. Else, if some candidate(s) K2-4 are in the dictionary, use the highest-ranked dictionary member.

   [3.1.] However, consider sending the token to an annotator if K1 is identical to the input and K1 is overwhelmingly more probable than K2 under the character model.

4. Else, when nothing is in the dictionary, send to an annotator.

   [4.1.] However, if K1 is identical to the input, consider using the input form without sending to an annotator if the difference in probability between K1 and K2 is large enough.

That these particular settings emerged from the dev set provides some insight on both the corpus and the more general operation of the system we are using. Step 1, using K1 if it is in the dictionary, applies even when the original input token differs from K1 and is also in the dictionary. This introduces some error, but implements far more changes that turn out to be real corrections (such as changing *Olavur* to *Ólavur*). Step 1 is the only case (besides human annotation) in which an original token that is present in the dictionary may end up changed - if K1 is not in the dictionary but any K2-4 is, the original input token that is in the dictionary will be used for output. This reflects tensions between confidence in the character HMM and the original text, with the dictionary 'endorsing' both; evidently, the OCR quality is just bad enough that the character model's top choice is a better bet in this situation - but only if it is the character model's top choice, and not K2-4.

The human annotator's role is mainly to adjudicate error-prone cases between the character model, which has confidence in a best candidate decoding, and the dictionary, which (sometimes due to dictionary incompleteness, and sometimes due to real OCR error) shows no confidence in the original OCR token (Step 3.1) or in any other candidate as well (Step 4). In both Step 3.1 and 4.1, we find that the character probability difference between the top candidate and the next-best candidate, when the top candidate is identical in form to the original token, is a fairly good signal for whether the *correct* form of that token is also identical with its original form. The precise value used is given in Equation 1, the ratio of this difference to the probability of K1.

**Equation 1.** $$\frac{P(K1)-P(K2)}{P(K1)}$$

It turns out that the value of this K1-K2 difference ratio tends to approach 1 when K1 is identical to the original token and to the gold standard correct token.

In Step 3.1, the character model endorses the original form of the token, while the dictionary selects some other lower-ranked candidate K2-4 because the original form/K1 is not

in the dictionary. In the majority of these cases the lower-ranked dictionary member is correct; however, there are some tokens for which the original form is correct and switching to a candidate K2-4 will lead to highly undesirable introduced error in the corpus. Close to 3% of the corpus falls into condition 3.1, so it is not feasible to pass all such tokens to a human annotator. Instead, a high K1-K2 difference ratio (threshold >0.99, set on 70k dev set; threshold referred to as TH3.1 in Table 5 below) flags most of the tokens for which K1 is correct - and flags few enough false positives (for which K2-4 is correct) that annotation of all flagged tokens is still practical - so that introduced error is minimised with efficient human effort.

In Step 4 when absolutely nothing is in the dictionary, if the top candidate K1 differs from the original token then there is very little good information for the correction system to work with so all such tokens are sent to the human annotator. These are only about 0.6% of tokens in the corpus, and they can be a good source of progressive augmentation to the dictionary. For cases falling under 4.1 where K1 does have the same form as the original input token, the K1-K2 difference ratio can provide a useful signal saving human effort. As in Step 3.1, this ratio tends towards 1 when K1/input form is correct; therefore, if it is sufficiently high (threshold >0.95, set on 70k dev set; referred to as TH4.1 in Table 5) then the token can bypass the annotator and be left as its original form/K1.

Although this section has presented corpus-specific settings for the Faroese typewritten data we are correcting, the basic heuristic method is widely adaptable, subdividing the correction problem into token classes from observable attributes and estimating optimal output settings for each class from a dev set of parallel text.

### 3.5. Interface

Annotator judgements are collected in a simple text interface.

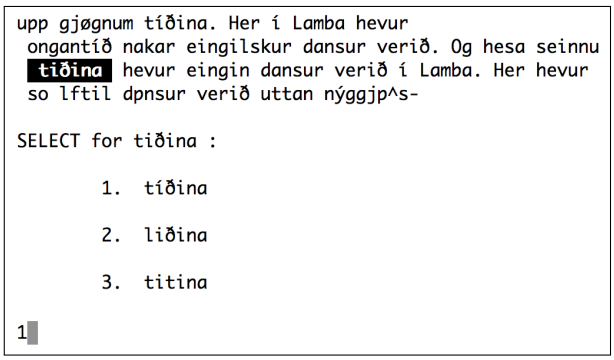


Figure 2: Interface for annotation.

The interface presents the target word highlighted in context, facilitating a quick and accurate decision. For each target word, up to three candidate corrections are offered; any of them can be chosen by number. The annotator can also type a single-character control code to keep the original word. If the original word is an error but none of the suggested candidates are correct, the annotator can type custom input. Font, size, and colours are easily controlled by users' general terminal interface settings.

Trials of this interface while producing parallel training/test data (further post-corrected by hand for gold standard) yield an estimated annotation speed of 4 seconds per target word, or 900 annotations per hour.

## 4. Results

We first present the primary evaluation of our correction system as developed in §3.4.1; then, three supplementary evaluations illustrate variation in performance as a function of annotator time, amount of HMM character model training data, and amount of heuristic-setting dev set data.

### 4.1. Main system

Applying the full correction procedure in four-fold cross validation with heuristics set on 10,000 words and tested on another 10,000 words of parallel text yields the projections for final corpus quality and human annotator cost shown in Table 3.

| | |
|---|---|
| Initial Error | 7.6% |
| Final Error | **1.3%** |
| Annotator Task | 2.2% |
| Introduced Error | 0.4% |
| Tokens Successfully Corrected | 6.6% |

Table 3: Current toolkit operation including annotation. All quantities are expressed by percentages of corpus tokens (words).

In this evaluation table, the Initial Error is the percent of tokens incorrect (different from gold standard) in the original input OCR, while Final Error is the total percent incorrect in the output. Annotator Task is the percent of the corpus that the annotator must give a judgement for. Introduced Error is the percentage of tokens in the corpus for which our automated system erroneously changes a form that was actually correct (matched the gold standard) in the original input. Tokens Successfully Corrected is the percentage of tokens that are real errors in the original input but have been corrected to match the gold standard, either by the automated system or by the human annotator.

Given the 2.2% annotator task and our estimate of 4 seconds per annotator judgement, the 2.7 million word Faroese corpus therefore requires about 65 annotator hours for full correction using the current version of our toolkit, compared with an estimated 2700 hours for manual editing without the toolkit.

| | |
|---|---|
| Initial Error | 7.6% |
| Final Error | **2.4%** |
| Introduced Error | 0.6% |
| Tokens Successfully Corrected | 5.7% |

Table 4: Operation of character model with dictionary and heuristic handling but no human annotation. All quantities are expressed by percentages of corpus tokens (words).

Even without human annotation, applying dictionary-informed heuristic conditional filtering to the *k*-best candidates performs much better than the plain dictionary-free HMM (final error 5.4%, see Table 2); Table 4 shows projections for this combination in direct comparison to Table 3.

### 4.2. Ablation analysis

Table 5 systematically illustrates the tradeoffs between annotator expense and corpus quality, for this Faroese corpus. The progression starts from the best fully automated annotator-free system (the same as in Table 4) as Stage 0, and builds at each stage by replacing the automated decision strategy of one more of the system's logically separable heuristic cases with a human annotator until reaching full human annotation in Stage 11. The ordering of stages in the ablation is somewhat arbitrary; we have attempted to give some illustrations of sensible choices at different annotation budgets, as well as pass through the system we actually use. Our chosen system (that of Table 3) is bolded; this reflects our target of around 2% annotator task.

The heuristic cases defining the ablation steps are those described in §3.4, created from logical combinations of four binary-valued attributes: E, whether the top-ranked generated candidate's form is Equal to the original input token; O, whether the Original token is in the dictionary; B, whether the Best generated candidate is in the dictionary; and K, whether any generated candidates K1-4 are in the dictionary. These attributes are listed under Switched in Table 5 to specify which heuristic has been switched to human annotation at each stage; their true/false values are coded +/-. Table 5 has 11 separable steps progressing from Stage 0, rather than only the 9 heuristic cases described in §3.4, because in our best-performing implementation two of the cases are further split by a threshold to conditionally send only some tokens to a human annotator. These split cases (using thresholds TH3.1 and TH4.1 in §3.4.1 above) are marked in the table alongside the EOBK attribute coding when a set of tokens falling over or under their threshold is switched to human annotation.

| Stage | Switched | Annotated | Final error |
|---|---|---|---|
| Stage 0 | | 0% | 2.41% |
| Stage 1 | E+O-B-K+>TH3.1 | 0.42% | 2.15% |
| Stage 2 | E-O-B-K- | 1.13% | 1.68% |
| **Stage 3** | E+O-B-K-<TH4.1 | **2.23%** | **1.34%** |
| Stage 4 | E-O+B+K+ | 2.50% | 1.27% |
| Stage 5 | E-O-B-K+ | 2.90% | 1.18% |
| Stage 6 | E+O-B-K+<TH3.1 | 5.77% | 0.79% |
| Stage 7 | E+O-B-K->TH4.1 | 7.46% | 0.57% |
| Stage 8 | E-O-B+K+ | 10.27% | 0.36% |
| Stage 9 | E-O+B-K+ | 10.49% | 0.35% |
| Stage 10 | E-O+B-K- | 10.55% | 0.34% |
| Stage 11 | E+O+B+K+ | 100% | 0% |

Table 5: Ablation analysis for amount of human annotation. All quantities are expressed by percentages of corpus tokens (words). The Annotated column lists how much of the corpus must be annotated by a human. The Switched column indicates which heuristic case has just been switched to human annotation at the current stage.

This exact ablation curve is specific to the unique corpus and initial heuristic settings it comes from, but it clearly illustrates a general principle for combining human annotation with an automated system: an annotator is effectively used in cases where they can annotate a manageable chunk of the data and have a beneficial impact with most of those annotations. Our system's automated generation of candidate corrections and heuristic case-based handling for them creates a structure that allows for an efficient use of the annotator's effort, and flexibility in decision-making depending on the annotation budget and corpus quality requirements.

### 4.3. Reduced character model training data

As described in §3.2, the character decoder that generates output forms is trained on 170,000 words of parallel text. Since this much parallel text could be expensive to produce if it is not already available independently, Table 6 shows the consequences of using character models trained on significantly less parallel training data, to observe the effect on system quality and usability. The first row is reproduced from Table 3, while results for the other two character decoders are reported under identical heuristic settings.

| Training Data | Corrected | Added Error | Annotated | Final Error |
|---|---|---|---|---|
| 170k | 6.6% | 0.4% | 2.2% | 1.3% |
| 100k | 6.6% | 0.5% | 2.5% | 1.4% |
| 50k | 6.3% | 0.4% | 2.6% | 1.6% |

Table 6: Performance when reducing character decoder training data: percent of true errors that are successfully corrected; percent of error introduced to originally correct tokens; percent of tokens judged by human annotator; total remaining error after processing and annotation.

Overall the character decoding component of our system appears fairly robust to major reduction in training data. Although some negative effect of the reduction is certainly evident, especially in the training set of 50,000 words, performance seems to degrade slowly.

The training size of 100k words appears to have a little more introduced error, and a higher workload for the annotator, when tested with the same heuristic settings that we found to be best when using full-size training data. Some of the increase in annotator task could be because the thresholds that control which tokens are annotated were set on a rather different dev set, and may not be well calibrated for this character model's probabilities. It would be possible to recalibrate them to reduce the annotation task, in exchange for raising the final error rate. However, another part of the increased annotator task is simply that more tokens overall fall into the annotator-directed heuristic cases. This is essentially because when this slightly degraded character decoder now fails to generate the right form (that is usually in the dictionary), it often instead generates forms that are not in the dictionary, and it is mainly cases lacking dictionary membership that fall to the annotator (§3.4.1). The summary effect of the 100k training set is perhaps to shift the system's entire annotation ablation curve (Table 5) to

a slightly more expensive place: very good quality is still achievable with a bit more annotator time, or if annotator time is strictly limited then quality will be slightly worse than if using the 170k training set.
With 50k words of training data, there is a reduced ability to correct some of the errors in the corpus, and the annotator task has increased further due to unsuccessful generated candidates that are not dictionary members. With the most expensive annotation requirement and yet the worst result (final error), training on only 50k words clearly hurts system performance. However, it does still significantly improve the corpus without extremely unreasonable annotator effort, reducing error from an initial rate of 7.6% (Table 3), and therefore may be acceptable to some users.

### 4.4. Reduced heuristic dev set data

In §4.1 we used dev sets of 10,000 words of parallel text to set optimal decisions for the heuristic cases. Here, we explore the outcome of using smaller dev sets to set heuristic decisions, while still testing these decisions on 10,000 words to obtain reliable comparisons to the test sets of §4.1. Table 7 shows the projected outcomes of using heuristics derived from a variety of smaller dev sets.

| Dev Size (words) | Annotated | Final Error |
|---|---|---|
| 8,000 | 2.2% | 1.3% |
| 5,000 | 2.2% | 1.3% |
| 3,000 | 1.8% | 1.5% |
| 2,000 | 2.5% | 1.6% |

Table 7: Impact of smaller dev sets on annotator workload and final error.

Table 7 illustrates sample results only, giving an example from one dev set at each of the listed sizes; this shows what kind of outcome is possible but does not necessarily reflect average expected performance for dev sets of that size.
For the dev sets of 8,000 and 5,000 words, the optimal heuristics were identical to those found in §4.1. Therefore, there is no change in projected results. This gives further assurance that our current system is based on adequate dev data, and may be reasonably expected to generalise well to the rest of the corpus when it is corrected. This result also indicates that if the amount of parallel text were very limited, the dev set could be fairly small, under 10,000 words for this corpus.
With a 3,000-word dev set, there is one difference in the apparently optimal settings; human annotation is not used in one case, because the somewhat sparse information in this dev data was insufficient to clearly identify all good opportunities to take advantage of annotation. The result of these settings misses the mark a bit - it does not fully use our intended annotator budget of 2%, and so has a worse final error rate than might otherwise have been achieved - but it is still probably a decent heuristic setting at a different target point on an annotation/error tradeoff curve (Table 5).
Using 2,000 words, 4 of the 9 heuristic cases have different settings than in the original system. The damage to results is not catastrophic; changes to low-frequency cases (among those most affected by sparse data) have limited impact. Still, with an increase in both annotator effort and final error, the lack of sufficient data in the 2,000-word dev set is clearly harmful. Using a statistically misleading (too small to be representative) dev set has a risk of setting misfit parameters that waste annotator effort without achieving quality worth the effort. The estimates of how much of the corpus will be annotated also become less accurate when they are estimated on smaller dev sets, resulting in logistical trouble from more annotation burden than expected, or suboptimal corpus quality if not using the full annotation budget.

## 5. Discussion

### 5.1. Previous Work

There is limited precedent for the problem of post-processing OCR output to a high quality from physically degraded input in low-resource languages. Doush and Al-Trad (2016) use both Microsoft Word and Google's spelling suggestions to moderately improve OCR error in Arabic. Afli et al. (2016) use machine translation to repair French OCR; the translation system is trained on 90 million words of parallel text that is all hand corrected by 2-3 annotators, and achieves 1.7% word error rate. Kolak and Resnik (2002; 2005) use HMM-based post-processing like ours that is applicable to low resource languages, but human annotation is not incorporated and error rates remain too high for our goals with the Faroese corpus. More generally, there are a variety of noisy channel model analogues to our scenario, in OCR itself (Natarajan et al., 1999; Natarajan et al., 2009; Shlien and Kubotal, 1986), transliteration (Jiampojamarn et al., 2007), machine translation of closely related languages (Pourdamghani and Knight, 2017), multilingual part of speech tagging (Duong et al., 2014), spell checking (Li et al., 2012), and speech recognition (Gales and Young, 2008), among others. These applications all tend to differ from ours in that they use much more training data, or are satisfied with higher output error rates in tasks ranging from much harder than ours to rather similar.
Our toolkit is situated in the paradigm of human-assisted NLP. Combining statistical NLP with human judgement, supervision, or correction is common in preparation of small but valuable high-quality resources for theoretical linguistics research, such as annotating the Icelandic corpus of Rögnvaldsson et al. (2012). The methodology used by Ingason et al. (2014) shows the current state of Faroese NLP for historical corpora; they used a bilingual (Icelandic) approach to NLP resource development, together with time-consuming manual correction to parse a Faroese corpus of 50,000 words. With a larger corpus and simpler task of OCR correction, we have placed relatively greater emphasis on reducing human annotator time. An alternative human-assisted system developed for correcting OCR error in Yupik (Schwartz and Chen, 2017) may be preferable to our method in some use cases, although it would not be able to effectively correct our Faroese corpus. The method of Schwartz and Chen (2017) takes advantage of systematic restrictions on Yupik syllable structure to identify words containing OCR error (violating legal syllable shapes), and flag them for annotator attention. This does not require a dictionary or parallel training text; therefore, unlike our

method, it could be used in extremely low-resource settings with no expected performance loss. However, the approach of Schwartz and Chen (2017) critically depends on the typology of the language it is applied to, and whether that language has such a restricted set of permitted syllable structures (detectable from the orthography) that most OCR errors will violate them. Faroese, unlike Yupik, permits a fairly large number of syllable shapes; many OCR errors (such as *nógu* which should be *nógv*) create locally reasonable sequences that happen to not result in real whole words of Faroese, so that a dictionary provides a more useful signal here. Additionally, the system for Yupik does not propose candidate corrections for the errors it identifies, so the human annotator must address all of them. This would require several times more annotator effort than we intended for the Faroese corpus: 7.6% of tokens assuming perfect error detection, with more effort per token than expressing a judgement in one keystroke. The language resources (parallel text and dictionary) that our method requires enable it to generate corrections, and implement many of them with no human attention, to use annotator time efficiently in large corpora.

### 5.2. General Application for OCR Correction

Any use of our toolkit for OCR post-processing must find a balance between three types of resources: parallel training text for character error models and heuristic estimation, a dictionary or wordlist to help filter decoding output, and human annotation to adjudicate between the character model and dictionary. The Faroese corpus we correct has a good amount of parallel data for training and development, and a large high-quality dictionary; our goal was to bring the corpus to a very low error rate with only a little annotator effort. Either of our major language resources may not be accessible for other corpora our toolkit is used with, the initial error rate of corpora will vary, and corpus developers' ultimate goals and priorities may differ from ours. We intend for our illustrative analyses to provide prospective users with at least a first glimpse at how applicable our tools might be for their problem, given their particular available language resources, amount of annotator time, and size of corpus the annotator's time must scale to.

The supplementary evaluations exploring how robust the toolkit is to artificial limitations in resource availability showed that, although we had around 250k words of Faroese parallel text available for training and development (produced by human editors correcting portions of the Faroese corpus for various purposes over a period of years), it is still possible to substantially reduce OCR error in the corpus with under half or even one quarter this amount of parallel text. Performance is clearly best when more training data is available, but when it must be limited (if no corrected parallel text already exists), our toolkit can still be expected to improve the condition of a corpus with a fairly reasonable annotator workload.

The Faroese dictionary is from an open-source spell checking project; although Faroese is spoken by a small population, it is a living language whose speakers have one of the highest rates of computer/internet usage globally. The strategy of §3.4 relies heavily on a dictionary, and while this enjoys evident success with the large dictionary we use for Table 3, it may be a vulnerability if correction quality degrades rapidly with reduced dictionary coverage. We recognise that many extremely low-resource languages will not have comparable dictionaries readily available. Alternate sources may include scraping Wikipedia or other websites, smaller wordlists created for any purpose, and the set of words in the training data. Requirements for useful dictionary size will also vary by language typology. In cases of relatively low dictionary coverage, we would suggest an iterative approach, building a larger dictionary with annotator-approved forms as corpus correction proceeds. Our method appears generally well suited to a bootstrapping approach, to progressively build up both the dictionary and a larger bank of parallel training/development text when needed: the initial cost to start up can be reduced because the method has at least reasonable performance with relatively lower amounts of input resources, and the subideal system can then be used to fairly quickly produce more corrected (parallel) text and augment the dictionary, creating the resources necessary to more accurately and efficiently correct large corpora.

### 5.3. Aligner Uses

Independently of our toolkit as a whole, the aligner can be broadly useful. It produces quality word-aligned data from parallel source texts where token count, tokenisation, line breaks, etc. may not agree. Texts produced by human editing will often not align straightforwardly with their sources; on the other hand, constraining editors to strictly preserve the token structure of input files in their corrections would increase their workload while degrading the natural language quality of the edited text.

Our aligner resolves this tension, and it may be helpful in a variety of noisy channel model NLP tasks (e.g. transliteration (Jiampojamarn et al., 2007)), or any setting with parallel text where at least one side has been produced by a human editor. Because the aligner algorithm is not dependent on external resources like a dictionary or human annotator, it can be applied to new corpora with confidence.

## 6. Acknowledgements

We thank Caroline Heycock, Tony Kroch, and the faculty of Føroyamálsdeildin for their support in the development of Faroese corpus resources; and Marine Carpuat and Sharon Goldwater for their technical insight. This work was supported by NSF grant DGE-1321851, and by DARPA Lorelei under Cooperative Agreement No. HR0011-15-2-0023.

## 7. Bibliographical References

Afli, H., Barrault, L., and Schwenk, H. (2016). OCR Error Correction Using Statistical Machine Translation. *International Journal of. Computational Linguistics and Applications*, 7(1):175–191.

Doush, I. A. and Al-Trad, A. M. (2016). Improving post-processing optical character recognition documents with arabic language using spelling error detection and correction. *International Journal of Reasoning-based Intelligent Systems*, 8(3-4):91–103.

Duong, L., Cohn, T., Verspoor, K., Bird, S., and Cook, P. (2014). What Can We Get From 1000 Tokens? A Case Study of Multilingual POS Tagging For Resource-Poor Languages. In *Proceedings of EMNLP*, pages 886–897. Association for Computational Linguistics.

Galbraith, D. (2016). Faroese ballad meter: a constraint-based account. In *NordMetrik: Versification, Metrics in Practice*, 5/25/16 - 5/27/16. University of Helsinki.

Gales, M. and Young, S. (2008). The application of hidden markov models in speech recognition. *Foundations and trends in signal processing*, 1(3):195–304.

Heycock, C. and Wallenberg, J. (2013). How variational acquisition drives syntactic change: The loss of verb movement in Scandinavian. *Journal of Comparative Germanic Linguistics*, 16(127).

Ingason, A. K., Loftsson, H., Rögnvaldsson, E., Sigurðsson, E. F., and Wallenberg, J. C. (2014). Rapid Deployment of Phrase Structure Parsing for Related Languages: A Case Study of Insular Scandinavian. In Nicoletta Calzolari, et al., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 91–95, Reykjavik, Iceland, May. European Language Resource Association (ELRA).

Jiampojamarn, S., Kondrak, G., and Sherif, T. (2007). 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. Association for Computational Linguistics.

Kleinberg, J. and Tardos, É., (2006). *Algorithm design*, chapter 6.6 Dynamic programming: Sequence alignment. Pearson Education.

Kolak, O. and Resnik, P. (2002). OCR error correction using a noisy channel model. In *Proceedings of the second international conference on Human Language Technology Research*, pages 257–262. Morgan Kaufmann Publishers Inc.

Kolak, O. and Resnik, P. (2005). OCR post-processing for low density languages. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 867–874. Association for Computational Linguistics.

Li, Y., Duan, H., and Zhai, C. (2012). A generalized hidden markov model with discriminative training for query spelling correction. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 611–620. ACM.

Natarajan, P., Bazzi, I., Lu, Z., Makhoul, J., and Scwhartz, R. (1999). Robust OCR of degraded documents. In *Document Analysis and Recognition, 1999. ICDAR'99. Proceedings of the Fifth International Conference on*, pages 357–361. IEEE.

Natarajan, P., MacRostie, E., and Decerbo, M. (2009). The BBN byblos Hindi OCR system. In *Guide to OCR for Indic Scripts*, pages 173–180. Springer.

Pourdamghani, N. and Knight, K. (2017). Deciphering related languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2503–2508.

Rögnvaldsson, E., Ingason, A. K., Sigurðsson, E. F., and Wallenberg, J. (2012). The Icelandic Parsed Historical Corpus (IcePaHC). In Nicoletta Calzolari, et al., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1977–1984, Istanbul, Turkey, May. European Language Resource Association (ELRA).

Schwartz, L. and Chen, E. (2017). Liinnaqumalghiit: A web-based tool for addressing orthographic transparency in St. Lawrence Island/Central Siberian Yupik. *Language Documentation & Conservation*, 11:275–288.

Shlien, S. and Kubotal, K. (1986). Optical character recognition of touching characters. In *Graphics Interface 1986*, pages 390–395. Canadian Information Processing Society.

Thráinsson, H., Petersen, H. P., Jacobsen, J. í. L., and Hansen, Z. S. (2012). *Faroese: An overview and reference grammar*. Føroya Fróðskaparfelag, Tórshavn.

## 8. Language Resource References

Føroyamálsdeildin. (n.d.). *Bandasavnið*. Fróðskaparsetur Føroya ('The tape archive, of the Faroese Language faculty, University of the Faroe Islands'), https://setur.fo/gransking/soevn/bandasavn/.

J. S. Andersen. (2001). *Føroyska orðalistin til rættlestur*. The Faroese Linux User Group, http://fo.speling.org.