OCR Error Detection and Post-Correction with Word2vec and BERTje on Dutch Historical Data

Nynke van ' t Hof[1,2], Vera Provatorova[1,3], Mirjam Cuper[2], and Evangelos Kanoulas[1]

[1]University of Amsterdam
[2]KB, National library of the Netherlands
[3]KNAW Humanities Cluster

High-quality Optical Character Recognition (OCR) output is essential for enhancing the accessibility of documents and enabling various Natural Language Processing (NLP) tasks. However, historical documents often present significant OCR errors due to their condition and variability in fonts and spellings. This study addresses the post-processing correction of OCR errors in Dutch historical documents, comparing the effectiveness of two widely used word embedding models: static word2vec and contextual BERT. Although BERT has shown promising results in previous studies, the reasons for its superior performance remain unclear. This research aims to identify the specific challenges faced by word2vec and determine if BERT can mitigate these issues more effectively. By analyzing both models in detail for error detection and correction tasks, we provide insights into their respective strengths and limitations. Our findings suggest that while both models have their merits, BERT demonstrates a notable advantage in handling some of the complexities of historical data, making it a valuable tool for OCR post-correction in digital humanities.

Keywords: OCR post-correction, Natural Language Processing, Word Embedding Models, historical data, digital heritage

1 Introduction

In digital humanities, particularly in the context of historical document analysis, the improvement of Optical Character Recognition (OCR) technology is of great importance. With a high quality of OCR-output, documents become more accessible to readers, and Natural Language Processing (NLP) tasks can thrive on the data. However, the extent to which all this is possible is dependent on the quality of the OCR-output. OCR on historical data often creates significant errors due to the poor condition of documents and variations in font size and spelling.

In this context, our research addresses the critical need for post-processing of OCR-generated machine-readable text. Specifically, our focus is on historical documents written in Dutch. We employ word embedding models (WEMs) as a key component to tackle this task effectively. WEMs have been shown to be promising for the task of post-correcting OCR-outputHämäläinen and Hengchen (2019); Nguyen et al. (2020); Pal and Mustafi (2020). They can be divided into static word embeddings and the more novel context-aware word embeddings. We turn our attention to two popular WEMs to represent them; static word2vec and context-aware BERTje (a Dutch version of BERT)Hämäläinen and Hengchen (2019); Nguyen et al. (2020).

Although alternative models might be interesting for this research, word2vec and BERTje have been selected for several reasons. FastText's character-level n-gram approach might be more effective than word2vec at capturing variations in historical spellings and surface errors. However, word2vec is a well-established model that offers a relatively straightforward implementation. Word2vec also serves as a common baseline for evaluating newer word embedding modelsDevlin et al. (2018); Pennington et al. (2014). For these reasons, it allows for a clearer comparison of BERTje's performance against a standard approach. Recently, GysBERT has been introduced as a strong contender against BERTje. Its training on historical data makes it a strong candidate for understanding the nuances of historical language, potentially leading to superior performance on the OCR post-correction task. However, at the start of our project, GysBERT was still in development. Additionally, comparing word2vec and BERTje, both non-specialized in historical data, provides more insight into the architecture and application of the models themselves. In contrast, using a specialized model like GysBERT would focus more on the influence of the data it was trained on. Therefore using BERTje instead of GysBERT can lead to clarifying insights.

Despite BERT showing promising results in OCR post-correction, existing studies lack an in-depth analysis of its performance when compared to previous models. This study aims to fill this research gap by thoroughly comparing BERTje and word2vec. Specifically, We compare the performance of contextualized word embeddings from BERTje and static word embeddings from word2vec in post-correcting OCR output from Dutch historical documents. By doing so, we aim to understand how different word embedding methods can benefit the processing of historical documents and improve the state-of-the-art.

This project is structured into three phases: first, we isolate the task of error detection; second, we conduct an isolated experiment on error correction; and third, we combine error detection and correction in a comprehensive evaluation process.

In this study, our primary objective goes beyond merely comparing the performance of BERT with word2vec in OCR post-correction. We also aim to delve into the underlying reasons for performance disparities, specifically by examining how each model addresses specific challenges; namely homonyms, historical spelling variations, out-of-vocabulary words, infrequent words, and real-word errors Wevers and Koolen (2020).

2 Background

2.1 OCR Post-correction

OCR post-correction involves three primary approaches, each with its set of merits and limitations Hämäläinen and Hengchen (2019); Nguyen et al. (2020); Salimzadeh

(2019).

Firstly, there's the crowdsourcing or manual error correction approach, which relies on people manually verifying and correcting OCR-generated output. While this method offers careful error correction, it is remarkably time-intensive.

The second approach, known as lexical error correction, employs a quicker process. It identifies and rectifies errors through dictionary-based comparisons. However, it struggles when confronted with words that are absent in the reference dictionary. This limitation becomes more pronounced when dealing with historical language, given the scarcity of relevant lexicons and the diverse linguistic nature of historical documents.

The third approach, statistical error correction, represents a diverse spectrum of methods, including machine translation and recurrent neural networks (RNNs). These methods are distinct in that they rely less on dictionaries. However, they may not fully capture nuanced linguistic differences in historical documents due to their statistical models being primarily based on modern language conventions. Word embeddings belong to this category.

2.2 Word embeddings

Word embeddings add a layer of semantics which is useful when dealing with historical data, known for its variability. This semantic layer enables independent text comprehension, which reduces reliance on statistical rules that are primarily based on modern language.

There are two main types of word embeddings: contextualized and context-free. Context-free or static models create a single embedding for each word, combining all possible senses. On the other hand, contextualized models, such as BERT, assign different vectors to different senses of the same word. Static WEMs like word2vec have some limitations, such as their data hunger, struggles with handling polysemy, and a tendency to rely on lexical approaches. They also face challenges with infrequent historical expressions and out-of-vocabulary words Hämäläinen and Hengchen (2019); Wevers and Koolen (2020); Wiedemann et al. (2019).

While static WEMs have found uses in tasks like lexical semantics, lexicalsemantic tasks, concept-based search, and (sentiment) classification, for OCR post-correction they have less often been appliedEgense (2017); Malte and Ratadiya (2019); Rezaeinia et al. (2019); Tulkens et al. (2016). However, Transformer models like BERT have already shown impressive performance in this contextNguyen et al. (2020). Previous comparisons between BERT and static WEMs have primarily focused on classification tasks. There has been limited exploration of the reason for BERT's effectiveness in OCR post-correction.

Several factors might possibly account for BERT's superior performance over static WEMs. BERT's heightened sensitivity to infrequent expressions makes it well-suited for smaller historical datasets. It effectively handles out-of-vocabulary words by breaking them into subtokens, enabling the understanding of less common historical expressionsWevers and Koolen (2020). For instance, when encountering the out-of-vocabulary word 'lantaarnopsteker' ('lamplighter', a word that will likely not be frequently used in modern documents), BERT identifies the subtokens that form the words 'lantaarn' and 'opsteker,'making it easier to grasp the word's meaning. This feature is particularly useful for Dutch, where composite words are often written as single tokens. BERT also addresses the challenge of polysemy by training separate vectors for different meanings of homonyms. Its training approach, involving the completion of masked words, aligns well with the task of OCR error correction, which will be

further explained in section 3.5. Lastly, BERT's bidirectional transformer architecture considers both the context on the left and the right of a word, setting it apart from its predecessor ELMo Wiedemann et al. (2019).

Additionally, we explore an intriguing phenomenon not previously explicitly recognized as a word2vec pitfall in the literature: the "real word error" (RWE), which occurs when an OCR machine generates an erroneous token that happens to be a valid word in the dictionary. For instance, consider the transformation of 'grass' into 'glass.' A dictionary-based method would fail to flag this as incorrect. Thus, part of the research goal revolves around the effectiveness of the discussed techniques in detecting and rectifying these unique tokens as well just mentioned, as well as real word errors.

However, BERT is not immune to limitations. It demands significant computational resources, making it less accessible for broader training on a diverse collection of datasets, which has led to historical datasets being more frequently overlooked. Moreover, due to its complexity and size, BERT lacks the transparency needed for understanding its inner workingsMalte and Ratadiya (2019); Wevers and Koolen (2020).

3 Experimental setup

3.1 The dataset

The dataset consists of the OCR-output of nearly 40,000 historical documents in Dutch. The documents are from the seventeenth up to the twentieth century, in five different categories: complete newspapers, individual news articles, books, book pages and typewritten radio bulletins. The dataset is freely accessible and in possession of the KB, National Library of the Netherlands, which additionally commissioned the creation of a ground-truth (GT) that is 99.95 percent accurateGT; Giovanni Colavizza (2021). The OCR has been performed by ABBYY. The data come along with multiple metadatasetsmet (a,b); Wilms et al. (2020).

For the purpose of analysing the efficiency of word2vec and BERTje at post-correcting specific tokens particularly, such as historical terms and homonyms, we used curated lists provided by the Dutch Language InstituteMol.

Computational limitations led to only using part of the data, with a proportional selection of the genres and centuries used.

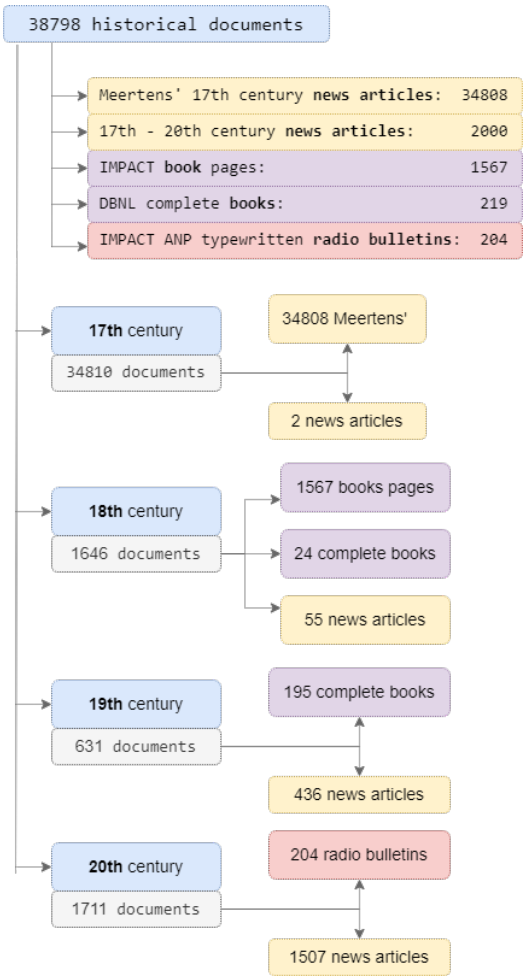


Figure 1: Distribution of historical documents in the KB Dataset categorized by century and genre

We conducted a thorough selection and sampling process to create a balanced and representative dataset. Initially, redundant files were removed to streamline the dataset.

For documents with multiple OCR outputs, the most recent output was selected to ensure the highest quality.

The dataset was further reduced in size due to computational constraints, aiming to take half a page from each document on average. This decision was based on the average length of a book page (272.25 words) and the average number of characters per word (5, including punctuation)cha; (pseud. van H. Brandt Corstius.). This consistent metric was applied across all document types.

As some documents were shorter than the cutoff, the final dataset included 13,390 pages from the original 378,008 pages. To ensure even distribution across the centuries, we adjusted the amount of text taken from each document based on the total number of documents available per century. This method ensured that each century and document type was adequately represented, maintaining a balanced dataset that reflects the diversity of historical documents. Due to computational expense, the models were eventually evaluated on a random selection of five hundred of these adapted pages. The experiment could however easily be repeated with more time and computational power on the complete 13,390 pages.

3.2 Models

The Dutch word2vec used was trained on Belgian magazines, Wikipedia, websites, the Social Media Dataset, and the SoNaR corpus consisting of newsletters, press releases, books, magazines, and newspapersTulkens et al. (2016).

Like BERT, BERTje is trained for two tasks: masked language modelling and next sentence prediction. It uses the same architecture and parameters as BERT. It is equivalent to BERTbase with 12 transformer blocks. It is more extensively trained on Dutch data than the multilingual BERT. Like Dutch word2vec, it has been trained on Wikipedia, the SoNar corpus, websites, but also books, including historical novels de Vries et al. (2019). The soft masked BERT models used for OCR post-correction, as well as the word2vec applications, are often applied together with other (deep learning) techniques that boost the performance. However, since this study is trying to compare the word embeddings themselves in their 'natural' state and is not focused on boosting performance, other models are not applied here.

As a baseline, we employ a lexical approach, using a dictionary to detect and rectify errors. This decision is consistent with established practices for OCR post-correction, as described in section 2 'Background'.

3.3 Preprocessing

The optimal preprocessing decisions for post-correction of OCR output are still under discussionKhirbat (2017). This is crucial, as it can significantly influence the entire process and its outcomes. In this project, preprocessing steps were applied to both the ground truth and the OCR output to enable a meaningful comparison. The following decisions were made regarding preprocessing:

- Punctuation was selectively removed, with exceptions made for full stops and hyphens to preserve hyphenated words. Full stops were retained to distinguish sentences from each other but were later removed during the evaluation phase. Although characters like question and exclamation marks could also serve as sentence separators, they were deemed unnecessary, as the primary objective was to segment the text into manageable portions.

- All capital letters were converted to lowercase.
- Lemmatization was intentionally avoided for the correction process, as advised by Khirbat in the literatureKhirbat (2017). Lemmatization would potentially leave out crucial structural information. For example, if 'has' were lemmatized to 'have,' the word embedding model would lose the context necessary to determine whether it should be preceded by 'they' or 'he' for example. These nuances are essential in the context of text correction with statistical methods.

The training phase encompassed all data combined, with no specific century-wise training, but for the last task the data were categorized by century. For this task of performing error detection and correction in a continuous pipeline, the results were provided by century to better analyse the efficiency of the models on older versus newer historical data. An alternative would be categorization by genre, but as a major challenge for OCR post-correction is the drastic change of language over timeNguyen et al. (2020), the division of the data was based on this aspect.

3.4 Alignment of the OCR-output and the ground truth

An essential preliminary task for evaluating the OCR post-correction output is aligning the ground truth with the OCR output. This has not yet been done in the provided dataset. The alignment process comprises of two steps: initial sentence-level alignment followed by word-level alignment.

To align sentences from the ground truth with the OCR output, we use approximate string matching for two primary purposes: First, it addresses situations where the ABBYY software missed complete sentences, parts of sentences, or entire pages, resulting in missing OCR output. Consequently, the ground truth for these missing sentences and pages is also removed. The calculation of the Word Error Rate (WER, the percentage of words transcribed incorrectly during the OCR process) and Character Error Rate (CER, the percentage of characters transcribed incorrectly during the OCR process), which indicate OCR output quality, occurs only after this step. This approach ensures that the WER reflects only the impact of OCR post-correction, without the added effect of redundant ground truth removal. The second purpose of approximate string matching for sentence alignment is to facilitate word-level alignment.

For the word-level alignment, each word in the OCR-output is aligned to its counterpart in the ground truth. This way, it can be determined whether an OCR token is an error and if a suggested correction is indeed correct, see fig. 2.

Figure 2: Example of word alignment between ground truth and OCR output

The word alignment in this study is based on the Fast Recursive Text Alignment Scheme (RETAS), which aligns texts by recursively cutting them into smaller partsYalniz and Manmatha (2011). The texts were first divided into sentences and then aligned word by word from both directions (front to back and back to front). Unlike RETAS, this method uses anchor words—unique and less frequent words—as reference points for alignment. Due to errors and missing words in the OCR output, approximate string matching was employed. An alignment checker was developed

to ensure the quality of alignment by warning about bad alignments and excluding them from evaluation. This checker was validated and found to be highly accurate, with precision, recall, and F1-scores all above 0.95. Eventually, the word alignment looks as in Figure 2. The bad alignments are removed for the evaluation using the forementioned alignment checker. The code for both the word alignment and the alignment checker can be found through the link provided in the appendix.

3.5 Approach

As mentioned in the introduction, this project is structured into an isolated detection task, an isolated correction task, and a final experiment where error detection and correction are combined consecutively, thus not in isolation. This approach follows a two-step process for OCR correction, as suggested by Schaefer et al. (2020), which enables us to reduce the likelihood of erroneously correcting OCR-output that is already correctSchaefer and Neudecker (2020). Additionally, this approach enhances our ability to pinpoint the precise stages, either the detection stage or the correction stage, where errors occur in the post-correction process. Furthermore, it allows us to fine-tune our strategies for each distinct stage, tailoring them based on each stages' specific requirements.

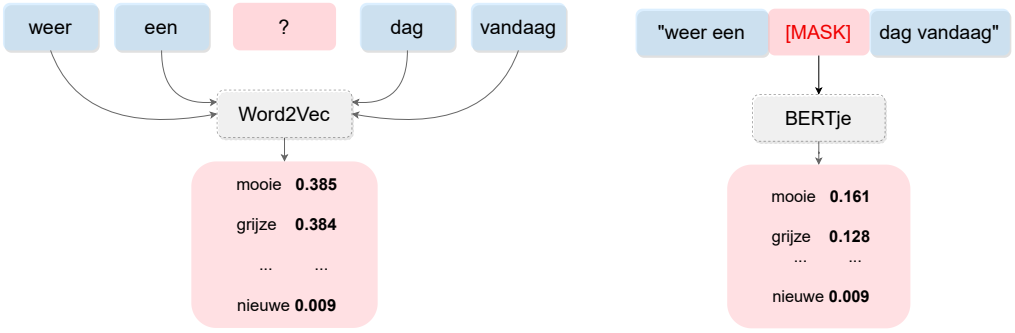
3.5.1 Detection task

To begin with, it is crucial to highlight that the detection task is performed on all tokens, both correct and incorrect. This ensures that the models are assessed on their ability to correctly identify errors without falsely labeling correct tokens.

The detection for the dictionary method, which is the baseline for this project, is the most straightforward: if a token is not in the dictionary, it is considered to be an error.

Many word2vec detection and correction tasks are as well reliant on a dictionary. Word clusters of words that are similar to the OCR token are created and the correct spellings are selected with a dictionary. Proposed methods that avoid this dependency do yet not generate a higher accuracy than those that do notHammarström et al. (2017). This problem will not be solved in the scope of this project, but since dependency on lexicons was mentioned as one of the pitfalls of word2vec OCR post-correction models, a lexicon will not be used. Alternatively, a method closer in approach to BERT's error detection process, namely Continuous Bag of Words (CBOW), is employed, as illustrated in Figure 3. This choice simplifies the process of comparing our approach to BERT's methodology.

In the CBOW version of word2vec, context words are used as input. The output consists of words that are semantically similar, determined by cosine similarity. The length of the list of predicted tokens can be specified with a parameter. The amount of input words is determined by the window size; in this project of size five. A larger window size (greater than 15) results in the model generating output words closely related in semantics to the input. On the contrary, a smaller window size (less than 15) yields tokens that are interchangeable, such as synonymsRong (2014); Yang (2024). For this project, a smaller window size of five is selected to ensure the model suggests words that could co-occur with the input words, see fig. 3a.



(a) Four context tokens used by word2vec to predict a new token based on cosine similarity

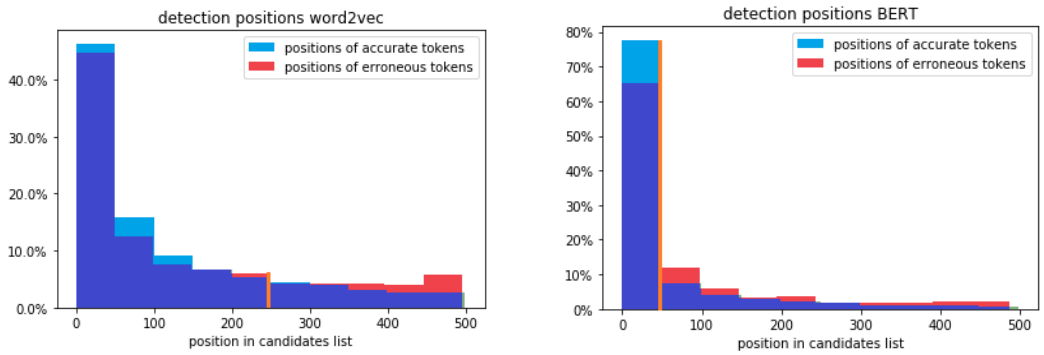
(b) A sentence used by BERTje to predict the missing token based on probability

Figure 3: Token prediction methods used by Word2Vec and BERTje

The same principle is applied with BERTje. However, instead of a window size with context words, the whole sentence is used as context, as BERT is trained and finetuned on sentences. The predicted candidates to fill the mask are ordered based on probability instead of cosine similarity. See fig. 3b.

The positioning of words within the list of output words can serve as an indicator of whether they are likely to be errors. Take the example from Figure fig. 3: "what a ??? day today". The word 'mooie' ('beautiful'), which appears as the top word in the suggestions to fill in the blank, is correctly spelled. However, if it were a misspelling, it would likely have rarely occurred in the provided context during pretraining and fine-tuning, resulting in its appearance at a lower position in the list. Therefore, the relative position of an OCR token in the candidates' list can be used to gauge whether it is an error or notHu et al. (2020).

However, where should the line in this list between likely correct tokens and likely erroneous ones be drawn? Determining the optimal parameter for the cutoff position, which signifies whether a token is an error, is performed through validation set analysis. It should be positioned at the transition point between the higher rankings in the candidate list, where mostly accurate tokens are found, and the lower rankings, where mostly erroneous tokens reside. While related literature often finds optimal parameters by running the model multiple times with various parameters[15], this approach can be computationally intensive. In contrast, by plotting the positions of correct and erroneous tokens, the optimal parameter can be found more conveniently, see figure fig. 4.



(a) The ideal length of the candidates list for detection from the validation set for word2vec.

(b) The ideal length of the candidates list for detection from the validation set for BERTje.

Figure 4: Distribution of accurate and erroneous tokens in the candidate lists generated by Word2Vec and BERTje during validation

The optimal detection parameters are ideally positioned where you can observe the orange lines in Figure 4. In the upper ranks of the candidates list, to the left of the orange line, the likelihood of encountering accurate tokens is higher than that of encountering erroneous ones. Conversely, to the right of the orange line, further down the candidates list, the tokens are predominantly erroneous, such as misspelled words. Thus, the orange line serves as the threshold for determining whether a token is likely to be erroneous; if it falls below this position in the list, it is considered likely erroneous and, hence, an error.

From figure fig. 4, a cutoff parameter of 250 is chosen for word2vec, while BERT performs best with a setting of 50. These values will be employed for the detection task. However, to gain insights into the impact of a more cautious parameter setting on the overall task, a value of 500 is retained for the complete task. This reduces the likelihood of the model unintentionally changing correctly spelled words into incorrect ones by mistakenly labeling them as errors.

3.5.2 Correction task

The isolated correction task, as mentioned, was performed solely on the tokens considered to be erroneous according to the gold truth dataset.

For correction, a similar candidate list method as the one used for detection is employed. Again, the validation set is used. This time, the optimal correction candidate for the error found in the validation set's ground truth is selected from the list. Since detection and correction are distinct tasks, a separate optimal parameter needs to be determined. This parameter determines how many correction candidates are considered. Generally, a higher value for this parameter results in better model performance as the actual ground truth correction is more likely to be on the candidates list, but it also increases computational time as more candidates have to be created and considered.

To identify the most effective parameter, a plot is generated to determine the typical position of correct corrections (ground truth tokens) within the candidates' list. For the model to perform well, it is crucial to include as many correct tokens as possible in the candidates list. If a ground truth token is not present in the list, the model cannot provide the correct correction. Figure fig. 5 displays the positions of correct tokens within the candidates lists generated by word2vec and BERT in the validation set. The

visualization extends up to position five hundred, which is the maximum value used in the literatureHu et al. (2020). Positions beyond five hundred are unattainable and are recorded as position five hundred in the plot.

Evidently, approximately 80% of the ground truth tokens are further down the candidates list than position five hundred. Consequently, the "topn correction" parameter should be substantially higher than five hundred. Ideally, improving the model's ability to recommend the correct candidates, such as by more extensive finetuning, would be the optimal solution. However, for the time being, the parameter is set to five hundred for both word2vec and BERTje.

Now how will the correction process be conducted? Let's begin by revisiting the straightforward baseline method. The baseline method relies on a dictionary to rectify erroneous tokens by replacing them with words found in the dictionary. This correction process is accomplished using the Levenshtein distance (LD), which measures the minimum number of character edits (insertions, deletions, or substitutions) required to transform one word into another. Given that OCR errors often result from character editsSalimzadeh (2019), the LD is a suitable choice. However, in this study, a Levenshtein ratio is employed, derived from the LD. To illustrate why, consider the LD between "now" and "not" as well as between "nowadays" and the misspelled "nowadayf." Surprisingly, the LD is identical for both pairs of words. However, when assessing the context, "nowadayf" is more likely to contain an error than "not". The Levenshtein ratio thus considers the length of the tokens as well.

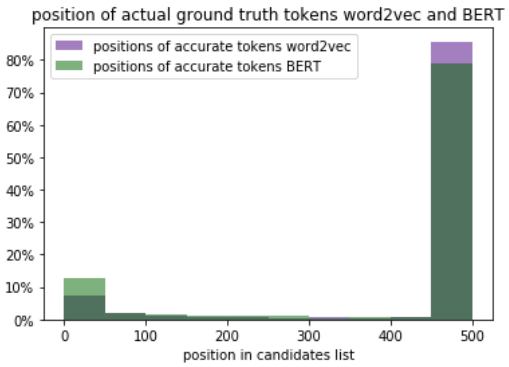


Figure 5: Positions of correct tokens (ground truth tokens) in the candidate lists generated by Word2Vec and BERTje from the validation set

When considering the Levenshtein ratio, a value of one indicates an exact match between tokens, while zero implies no similarity whatsoever. In the baseline method, the predicted correction is determined by selecting the word from the dictionary that, when compared to the OCR token, have the highest Levenshtein ratio. However, due to computational constraints, this ratio is only computed for words that possess the same character count as the OCR error or at most one character less or one character more. This broader range accounts for potential errors resulting from insertions or deletions.

Sorting method			
	Levenshtein ratio	Cosine (word2vec)	Probability (BERTje)
"nieuwe"	0.91	**0.179**	**0.161**
"nieuws"	0.91	0.051	0.128
"mooie"	0.40	0.385	0.009
"grijze"	0.36	0.384	0.001

Figure 6: Method for selecting the optimal correction candidate. Candidates are first sorted based on the Levenshtein ratio and then further sorted based on likelihood (cosine similarity for word2vec or probability for BERTje) from the context

For the correction with WEMs, there are two approaches of context sensitive text correction with WEMs mentioned in the literatureFivez et al. (2017); Hu et al. (2020). The first approach relies on a lexicon, which, as previously noted, is a pitfall this project tries to avoid. The other uses a list of candidates as was previously described. This method will be used in this project. The selection of the right candidate for correction is based on the Levenhstein ratio and also a cosine similarity for word2vec and a probability for BERT.

Typically, two primary methods are employed for this candidate selection task. These approaches have been compared in this study to determine the most suitable one for application. The first involves sorting candidates based on the Levenshtein distance, followed by considering either cosine similarity or probability. However, this method exhibits a bias towards frequently occurring words. For instance, "van" (meaning "of") is often suggested. This is especially done by word2vec, because "van" frequently appears in various contexts, and word2vec lacks grammatical structure indicators like BERTje. Consequently, in the case of word2vec, there's some debate in related literature regarding the removal of stopwordsHämäläinen and Hengchen (2019); Kim et al. (2019).

The other approach to this is the 'calculation method,' where a metric is calculated to determine the sorting order. A pre-existing calculation derived from literature, where the highest 'score' provides the best correction candidate, goes as follows[12]:

$$score = normalized\,cos\,or\,prob/(1 - LR)$$

The candidate with the highest score is selected as the correction. In this study, the Levenshtein ratio is used. However, unlike the Levenshtein distance, where a lower value indicates a close match, a high LR indicates a close match. An LR value of one means an exact match. Therefore, (1 - LR) is employed in this context. Additionally, to ensure a fair comparison between word2vec and BERT, the cosine similarity and probabilities were normalized for more consistent calculations.

These ranking methods was experimented with on the validation set. As the 'sort method without removing stopwords' performs best for both techniques on the validation set section 7, this method will be used on the test set. This means stopwords are removed from the candidates list and correction candidates will be sorted based on the Levenshtein distance first and subsequently by either cosine similarity or probability.

3.5.3 Complete OCR post-correction task

Instead of now performing detection and correction in isolation, the whole process is combined in this last task. This is done a fluent pipeline by first performing the

detection task on both the correct and the erroneous tokens, then feeding the tokens detected as errors to the correction task.

3.6 Evaluation

In the evaluation of the detection task, metrics such as recall, precision, and F1-score are employed, complemented by accuracy. We chose recall, precision, F1-score, and accuracy to provide a comprehensive evaluation of the models, capturing both the quality of the positive predictions and the overall correctness of the results. The metrics are chosen to evaluate true negatives and false positives on correct tokens as well as false negatives and true positives on erroneous tokens. This thorough evaluation ensures that both the accurate detection of errors and the avoidance of incorrectly labeling correct tokens are taken into account.

However, for the correction task, only accuracy is considered as this is a binary classification task. To comprehensively assess the overall method's performance, we calculate the CER and WER using the OCR post-corrected dataset. These metrics can be compared to the CER and WER from the pre-correction OCR output, as discussed in the preprocessing section.

As this project is mostly interested in the effect of the pitfalls of word2vec on both word2vec and BERTje's performance, the performance on these specific pitfalls is also measured using recall, precision and F1-score for the detection and accuracy for the correction task. These pitfalls were mentioned earlier to be homonyms, historical spelling variations, out-of-vocabulary words, infrequent words, and real word errors.

This analysis is also applied on the detection and correction of tokens that contain these special tokens in their context (a window for word2vec and the whole sentence for BERT). Something like a real word error or homonym in the context window could potentially mess up the detection and correction of a token as well, as wrong candidates might be suggested.

Thus the evaluation during the detection, correction, and combined task is applied on three levels: all available tokens, these mentioned special tokens like homonyms and real word errors, and words that have these special tokens in their direct environment.

4 Results

In bold are the F1 and accuracy scores of the methods that perform the best in each category (token or century) compared to the other methods. An asterisk indicates the F1 or accuracy scores that are lower than the average performance of that method on all tokens or all centuries combined. Scores marked with an asterisk thus suggest suboptimal performance compared to the overall average. For instance, in Table 1, the F1 score for the baseline method on historical tokens is 0.765, which is in bold because it is the highest among the methods. In contrast, the F1 score for word2vec on homonyms is 0.416, marked with an asterisk because it is lower than word2vec's average F1 score on all tokens, which is 0.424.

4.1 Detection task

Table 1 presents the error detection performance by token type. Table 2 shows the metrics on the detection task when certain tokens are in the model's context. Here the occurrence refers to the number of times the model has seen a specific token, like a homonym, in its context during the whole task. This occurrence value is different for

each model because word2vec uses a window, while BERTje uses the whole sentence as context. Table 3 shows the results by century.

The baseline method achieved strong overall performance. BERTje outperformed other methods in detecting Out-of-Vocabulary (OOV) and Real Word Error (RWE) tokens, and achieved exceptional overall detection.

Both WEMs showed a significant advantage over the baseline in detecting OOV and RWE errors, as expected for methods that go beyond dictionary-based approaches.

	special token:	**homonyms**	**historical**	**OOV**	**infrequent**	**RWE**	**none**	**all**
occurrence		9797	32679	differs	32450	32450	3510	38533
precision	baseline	0.869	0.927	0.453	0.792	0	0.901	0.784
	word2vec	0.576	0.419	0.750	0.417	0.917	0.071	0.433
	BERTje	0.222	0.185	0.686	0.186	0.976	0.129	0.733
recall	baseline	0.780	0.708	0.292	0.602	0	0.795	0.555
	word2vec	0.647	0.470	0.499	0.478	0.477	0.152	0.465
	BERTje	0.325	0.423	0.799	0.443	0.858	0.129	0.826
F1	baseline	**0.796**	**0.765**	0.330*	**0.654**	0	0.828	0.615
	word2vec	0.576	0.416*	0.516	0.420*	0.612	0.082	0.424
	BERTje	0.241*	0.236*	**0.708***	0.242*	**0.990**	0.129	**0.750**
accuracy	baseline	0.932	0.868	0.270	0.813	0	0.933	0.778
	word2vec	0.713	0.536	0.379	0.542	0.477	0.051	0.527
	BERTje	0.442	0.320	0.659	0.318	0.858	0.171	0.463

Table 1: Precision, Recall, F1-score, and Accuracy of the error detection task by token type

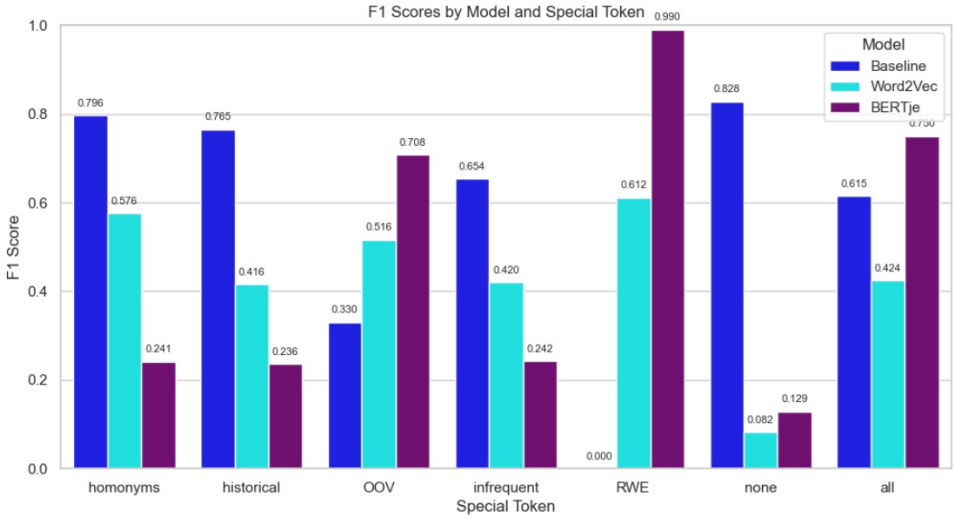


Figure 7: F1-scores on error detection task by token by model, , where 'all' stands for all tokens (special and non-special tokens together

Word2vec seems to consistently show a superior performance compared to BERTje when homonyms, historical words, OOV tokens and infrequent words are in its context.

	token in context:	homonyms	historical	OOV	infrequent	RWE	none	all
occurrence word2vec	→	20903	33395	9239	33460	4902	112	38533
occurrence BERTje	→	29561	32992	26241	38351	13122	695	38533
precision	word2vec	0.865	0.591	0.610	0.570	0.761	0.894	0.443
	BERTje	0.733	0.439	0.460	0.424	0.747	0.626	0.733
recall	word2vec	0.956	0.784	0.751	0.771	0.852	0.894	0.465
	BERTje	0.826	0.652	0.650	0.642	0.848	0.636	0.826
F1	word2vec	**0.878**	**0.635**	**0.645**	**0.617**	**0.785**	0.894	0.424
	BERTje	0.750	0.491*	0.505*	0.477*	0.772	0.629	**0.750**
accuracy	word2vec	0.744	0.676	0.667	0.659	0.538	0.071	0.527
	BERTje	0.463	0.419	0.422	0.412	0.478	0.099	0.463

Table 2: Precision, Recall, F1-score, and Accuracy of the error detection task for tokens in context

	by century:	1600s	1700s	1800s	1900s	all
occurrence	→	3949	2865	15318	15052	37184
precision	baseline	0.820	0.845	0.585	0.458	0.799
	word2vec	0.820	0.845	0.585	0.458	0.799
	BERTje	0.231	0.221	0.166	0.165	0.225
recall	baseline	0.623	0.311	0.182	0.218	0.586
	word2vec	0.465	0.434	0.478	0.488	0.465
	BERTje	0.456	0.794	0.806	0.798	0.497
F1	baseline	**0.683**	**0.421***	**0.269***	**0.260***	**0.643**
	word2vec	0.453	0.258*	0.207*	0.168*	0.424
	BERTje	0.292	0.327	0.262*	0.242*	0.290
accuracy	baseline	0.7701	0.807	0.829	0.853	0.778
	word2vec	0.549	0.392	0.349	0.336	0.527
	BERTje	0.308	0.532	0.560	0.613	0.340

Table 3: Precision, Recall, F1-score, and Accuracy of the error detection task by century

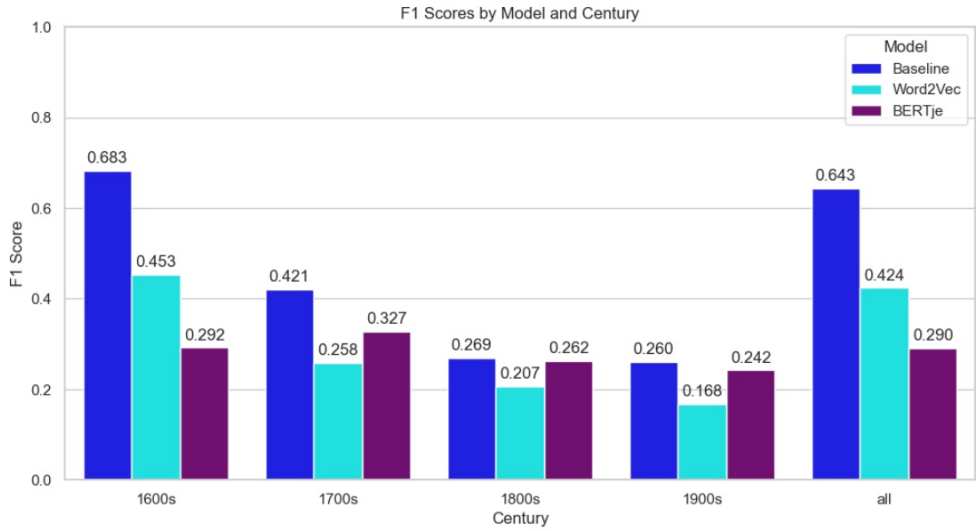


Figure 8: F1-scores on error detection task by century, where 'all' stands for all centuries together

Surprisingly, all methods showed better detection rates on older documents compared to newer ones. Especially the baseline dictionary-based method outperforms both word2vec and BERTje in texts from the seventeenth and eighteenth century at the detection task when considering all tokens from these centuries' datasets, not just the special tokens.

4.2 Correction task

Table 4 shows the results on the error correction task by token type. In contrast to the other tokens, only the occurrence of out-of-vocabulary words witnessed by the models differs for each model, as they each use their own vocabularies. Table 5 shows the results for tokens in context. Table 6 aggregates the results by century.

Word2vec excels in error correction for special tokens, despite its previously identified reliance on lexical methods as a potential weakness.

	special tokens:	homonyms	historical	OOV	infrequent	RWE	none	all
	occurrence →	656	3788	differs	3950	3510	differs	6772
baseline	accuracy	0.142*	0.333	0.397	0.337	0*	0.408	0.332
word2vec	accuracy	**0.581**	**0.618**	**0.436**	**0.647**	**0.442**	0.5	**0.593**
BERTje	accuracy	0.537	0.505	0.421*	0.509	0.414*	Nan	0.485

Table 4: Accuracy of the error correction task by token type

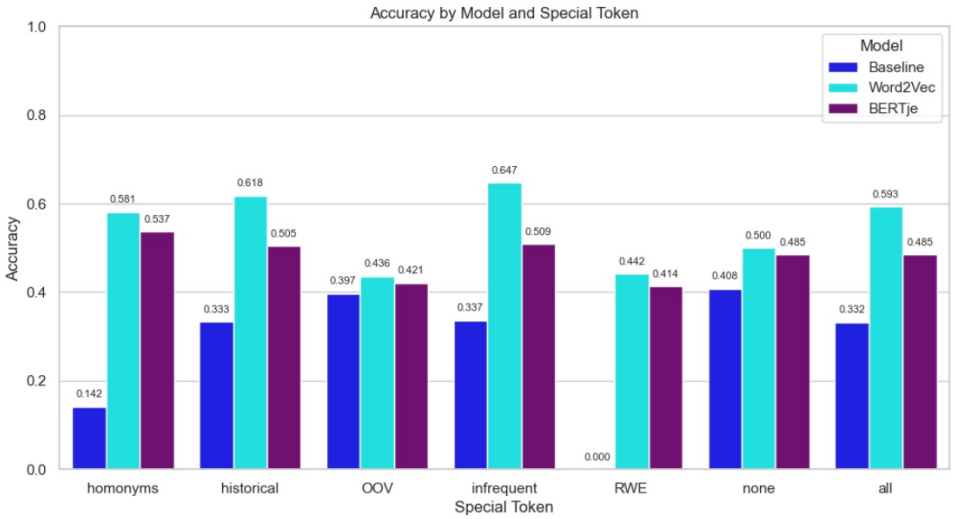


Figure 9: Accuracies on error correction task by token by model, where 'all' stands for all tokens (special and non-special tokens together)

	special tokens in context:	homonyms	historical	OOV	infrequent	RWE	none	all
	occurrence word2vec →	1569	3104	1873	3109	1533	80	6772
	occurrence BERTje →	2503	3222	2798	3228	2152	58	6772
word2vec	accuracy	**0.151**	**0.305**	**0.258**	**0.299**	**0.172**	0	**0.593**
BERTje	accuracy	0.100*	0.050*	0.048*	0.047*	0.052*	0.058	0.485

Table 5: Accuracy on correction task by token in context.

	by century:	1600s	1700s	1800s	1900s	all
	occurrence →	1666	728	2538	1702	6634
baseline	accuracy	0.362	0.225*	0.182	0.206*	0.332
word2vec	accuracy	**0.622***	**0.473***	**0.466***	**0.508***	**0.593**
BERTje	accuracy	0.494	0.468	0.426*	0.452*	0.485

Table 6: Accuracy on correction task by century.

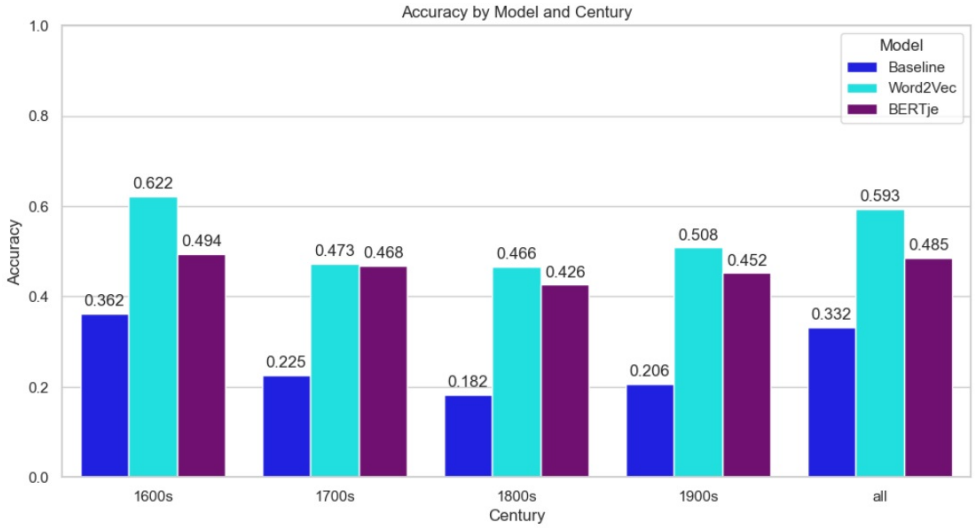


Figure 10: Accuracies on error correction task by century, where 'all' stands for all centuries together

4.3 Complete OCR post-correction task

Table 7 shows the result on the end-to-end OCR post-correction task. The Character Error Rate and Word Error Rate can be interpreted as percentages. An improvement of 40% however should not be interpreted as an improvement of 40 percent in regard to the old CER or WER, but as a decrease of 40 CER or WER expressed in percentages. As the improvement in Table7 appears negative, this means that the amount of errors after the end-to-end task has become worse. Word2vec and BERTje appear to perform at least less suboptimal than the baseline in this regard.

		old WER	new WER	improvement	old CER	new CER	improvement	total words
1600s	baseline	24.842	67.192	-41.190	7.187	51.320	-42.412	45505
	word2vec	24.842	56.644	**-28.999**	7.187	42.329	**-31.414**	45505
	BERTje	24.842	74.800	-48.528*	7.187	51.320	-42.412	45505
1700s	baseline	21.833	77.605	-55.251*	6.826	73.009	-65.555*	52316
	word2vec	21.833	61.671	-38.946*	6.826	47.880	-39.842*	52316
	BERTje	21.833	57.783	**-34.968**	6.826	45.922	**-37.839**	52316
1800s	baseline	14.278	61.520	-47.242*	5.670	72.972	-67.302*	128608
	word2vec	14.278	61.435	-47.158*	5.670	55.306	-49.636	128608
	BERTje	14.278	50.789	**-36.512**	5.670	50.637	**-44.967***	128608
1900s	baseline	13.224	51.844	-38.620	6.044	47.994	-41.950	114225
	word2vec	13.224	58.822	-45.598*	6.044	41.739	-35.695*	114225
	BERTje	13.224	43.596	**-30.372**	6.044	34.281	**-28.237**	114225
all	baseline	16.496	61.503	-44.772	6.176	61.710	-55.208	340654
	word2vec	16.496	59.955	**-42.948**	6.176	47.883	**-41.023**	340654
	BERTje	16.496	52.659	-35.821	6.176	44.520	-37.921	340654

Table 7: Performance on the complete OCR post-correction task by century

5 Discussion

5.1 Detection task

BERTje faced challenges with all types of special tokens. This seemed especially problematic when BERTje was applied to not the special tokens themselves per se, but sentences that contained those special tokens. This suggests BERTje might not have been fine-tuned sufficiently on historical data to accurately embed these tokens. Word2vec outperformed BERTje on the detection task involving special tokens in context words, indicating a potentially better alignment with such tokens. This might be explained by Word2vec's ability to create word clusters based on semantic similarity and its static nature, which can result in more stable representations for certain contexts. However, a significant part could also be due to BERTje's data-intensive nature, as it requires more data for effective training.

As noticed, all methods showed better detection rates on older documents compared to newer ones. While the baseline's success can be explained by its historical dictionary, further explanations for the other models are explored in the following paragraphs.

5.2 Correction task

Word2vec's strong performance on Real Word Error tokens challenges the notion that its dependency on lexical methods has to be detrimental. Regarding its performance on other special tokens, one possible explanation for the baseline's lower accuracy could be its lack of a tie-breaking strategy for instances where multiple candidates share the same Levenshtein ratio. Given its extensive dictionary, such ties are frequent, and the baseline essentially makes a random selection from among these candidates. This highlights the significant advantage offered by word embeddings. It is worth noting that BERTje may perform marginally worse than word2vec due to its potentially lesser familiarity with the dataset at this stage.

As mentioned in the results, during the correction phase, just as during the detection phase, all methods exhibit reduced performance on modern data. For WEMs, there is an interesting explanation for this trend. While attempting to correct modern text, BERTje consistently suggests contemporary equivalents for historical spelling

variations. This issue may also contribute to the detection challenges, as historical spelling variants could potentially be interpreted as errors.

5.3 Complete OCR post-correction task

When considering the post-correction task in its entirety, it becomes clear from the results that this task is a very complex endeavor. This underscores the importance of analyzing the correction and detection tasks separately. Since accurate tokens and characters outnumber erroneous ones (as indicated by the Word Error Rate and Character Error Rate), there is a higher likelihood of a correct token being mistakenly identified as an error if a model is not adjusted for this bias. This introduces the risk of erroneously correcting tokens that are already accurate, thereby increasing the WER and CER. In contrast, erroneous tokens are less common, making the chance of correcting them relatively smaller.

A major challenge with all these methods, as evidenced by the increase in CER and WER rates, is their tendency to make inappropriate corrections. The models often misidentify correct tokens as errors and alter them unnecessarily, leading to a net increase in errors. However, this does not mean that OCR post-correction is an unpromising endeavor. During this research, several pitfalls, such as the data hungriness of BERTje, have been identified. By addressing these issues, conducting more experiments, and combining the strengths of various models, it is possible that better results could be achieved.

5.4 Limitations

Due to the computational constraints of this study, a smaller dataset was used. This resulted in less fine-tuning, a less representative performance, and reduced evaluation materials. BERT may require more extensive fine-tuning to reach peak performance on historical data.

There are some smaller limitations of this research in the analysis. One problem with the analysis of the performance of the detection task on special tokens, is that its design is more fit for the analysis on the correction task. It is mostly considered whether the ground truth token is a special token, such as an OOV word, to see how often these can be accurately corrected. This is however less relevant for the detection since the detection task its focus is on the OCR output. There are more analysis tasks that should get a more fitting and in-depth analysis, as this study was merely a broad overview of many aspects of OCR-post correction.

6 Conclusion and future work

BERTje appears to perform the least suboptimally in the overall OCR post-correction task. In the separate detection and correction tasks, the dictionary method and word2vec alternatively yield the best results. Assessing BERTje's performance in detecting and correcting special tokens or words within their context does not definitively resolve whether BERTje can overcome the limitations of word2vec. However, it also cannot be ruled out.

Nonetheless, it is valuable to adopt a more expansive perspective and concentrate on the overarching objective rather than delving into the detailed analyses. This principle is particularly relevant when considering the application of these methods to historical data. This project aims to extend beyond the methods themselves and their potential

contributions to enhancing the state of the art. It also addresses the types of historical documents that could benefit from these techniques. This was done by looking at special tokens and different centuries, for example.

Highlighting the differences between data from diverse time periods would be a great contribution. Additionally, exploring whether fine-tuning on these periods separately or on all periods combined yields better results would be interesting. Future research could examine the effect of varying amounts of fine-tuning of word2vec and BERT on both modern data and different centuries. This would help determine the impact on any bias towards modern language. In this sense, this study serves as an invitation to embrace the challenges presented in this domain, which is an interdisciplinary task. This project is also an invitation to engage in this challenge.

While this study provides valuable insights into the performance of BERTje and word2vec on OCR post-correction for historical documents, several limitations must be acknowledged. Firstly, the research faced computational constraints, which led to the use of a reduced dataset size. This limitation affected the extent of fine-tuning that could be performed on the models. Additionally, the study did not include GysBERT, a model specifically trained on historical data, during its initial phase. The absence of GysBERT potentially limited the accuracy of the results. Moreover, the training data used for the models might have introduced a bias towards modern language, which could affect the generalizability of the findings to historical texts.

Future research should address these limitations by utilizing larger and more diverse datasets. Incorporating advanced models such as GysBERT, which is specifically designed for historical data, could enhance OCR post-correction accuracy. Additionally, exploring the impact of varying levels of fine-tuning on both modern and historical data would provide deeper insights into the models' performance. Specifically, investigating how different fine-tuning strategies on data from various centuries influence the results could offer significant improvements.

Recent advancements in the field, such as the use of transformers by models like ChatGPT, should be explored for OCR post-correction. GysBERT, a historical adaptation of BERTje, represents a promising avenue for future research. Evaluating its performance against the models used in this study could help address some of the limitations identified when working with historical data. By focusing on these areas, future studies can contribute to the development of more effective methods for processing historical documents.

References

Ground Truth KB. `https://lab.kb.nl/dataset/historical-newspapers-ocr-ground-truth`. URL `https://lab.kb.nl/dataset/historical-newspapers-ocr-ground-truth`.

Gigant-molex (version 1.0) (2019) [data set]. available at the dutch language institute: http://hdl.handle.net/10032/tm-a2-p9.

Tijdschrift Literatuur Zonder Leeftijd. Jaargang 14. URL `https://www.dbnl.org/tekst/_lit004200001_01/_lit004200001_01_0049.php`.

DBNL (2016), DBNL OCR data set. KB Lab: The Hague., a. URL `http://doi.org/10.5281/zenodo.3239290`.

IMPACT Project, IMPACT KB Ground-truth. KB Lab: The Hague. http://lab.kb.nl/dataset/ground-truth-impact-project, b.

Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. Bertje: A dutch bert model. *arXiv preprint arXiv:1912.09582*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Thomas Egense. Automated improvement of search in low quality ocr using word2vec, 2017. [Online; posted 2-February-2017].

Pieter Fivez, Simon Suster, and Walter Daelemans. Unsupervised context-sensitive spelling correction of clinical free-text with word and character n-gram embeddings. In *BioNLP 2017*, pages 143–148, 2017.

Mirjam Cuper Giovanni Colavizza. Is your OCR good enough? A comprehensive assessment of the impact of OCR quality on downstream tasks [dataset]. http://doi.org/10.5281/zenodo.4498186, 2021.

Mika Hämäläinen and Simon Hengchen. From the paft to the fiiture: a fully automatic nmt and word embeddings method for ocr post-correction. *arXiv preprint arXiv:1910.05535*, 2019.

Harald Hammarström, Shafqat Mumtaz Virk, and Markus Forsberg. Poor man's ocr post-correction: Unsupervised recognition of variant spelling applied to a multilingual document collection. In *Proceedings of the 2nd International Conference on Digital Access to Textual Cultural Heritage*, pages 71–75, 2017.

Yifei Hu, Xiaonan Jing, Youlim Ko, and Julia Taylor Rayz. Misspelling correction with pre-trained contextual language model. In *2020 IEEE 19th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pages 144–149. IEEE, 2020.

Gitansh Khirbat. Ocr post-processing text correction using simulated annealing (opteca). In *Proceedings of the Australasian Language Technology Association Workshop 2017*, pages 119–123, 2017.

Jeongin Kim, Taekeun Hong, and Pankoo Kim. Word2vec based spelling correction method of twitter message. In *Proceedings of the 34th ACM/SIGAPP symposium on applied computing*, pages 2016–2019, 2019.

Aditya Malte and Pratik Ratadiya. Evolution of transfer learning in natural language processing. *arXiv preprint arXiv:1910.07370*, 2019.

Thi Tuyet Hai Nguyen, Adam Jatowt, Nhu-Van Nguyen, Mickael Coustaty, and Antoine Doucet. Neural machine translation with bert for post-ocr error detection and correction. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, pages 333–336, 2020.

Aditya Pal and Abhijit Mustafi. Vartani spellcheck–automatic context-sensitive spelling correction of ocr-generated hindi text using bert and levenshtein distance. *arXiv preprint arXiv:2012.07652*, 2020.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Battus (pseud. van H. Brandt Corstius.). *Opperlandse taal-& letterkunde*. Querido, 1984.

Seyed Mahdi Rezaeinia, Rouhollah Rahmani, Ali Ghodsi, and Hadi Veisi. Sentiment analysis based on improved pre-trained word embeddings. *Expert Systems with Applications*, 117:139–147, 2019.

Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

Sara Salimzadeh. *Improving OCR Quality by Post-Correction*. PhD thesis, Universiteit van Amsterdam, 2019.

Robin Schaefer and Clemens Neudecker. A two-step approach for automatic ocr post-correction. In *Proceedings of the The 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 52–57, 2020.

Stéphan Tulkens, Chris Emmery, and Walter Daelemans. Evaluating unsupervised dutch word embeddings as a linguistic resource. *arXiv preprint arXiv:1607.00225*, 2016.

Melvin Wevers and Marijn Koolen. Digital begriffsgeschichte: Tracing semantic change using word embeddings. *Historical Methods: A Journal of Quantitative and Interdisciplinary History*, 53(4):226–243, 2020.

Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. Does bert make any sense? interpretable word sense disambiguation with contextualized embeddings. *arXiv preprint arXiv:1909.10430*, 2019.

L. Wilms, R. Nijssen, and T. Koster. Historical newspaper ocr ground-truth data set. kb lab: The hague., 2020.

Ismet Zeki Yalniz and Raghavan Manmatha. A fast alignment scheme for automatic ocr evaluation of books. In *2011 International Conference on Document Analysis and Recognition*, pages 754–758. IEEE, 2011.

Chaohao Yang. Learning word embedding with better distance weighting and window size scheduling. *arXiv preprint arXiv:2404.14631*, 2024.

7 Appendix

The code can be found at `https://github.com/VantHof-Thesis/OCR-post-correction/tree/main/all_code`.

7.1 BERTje specifics

BERTje specifics	
Optimizer:	Adam
Learning rate:	5e-5
Epochs:	4
Dropout:	0.1
Layers:	12 (BERT base)
Attention heads:	12 (BERT base)
Max_padding:	512 (BERT base)

7.2 Validation figures

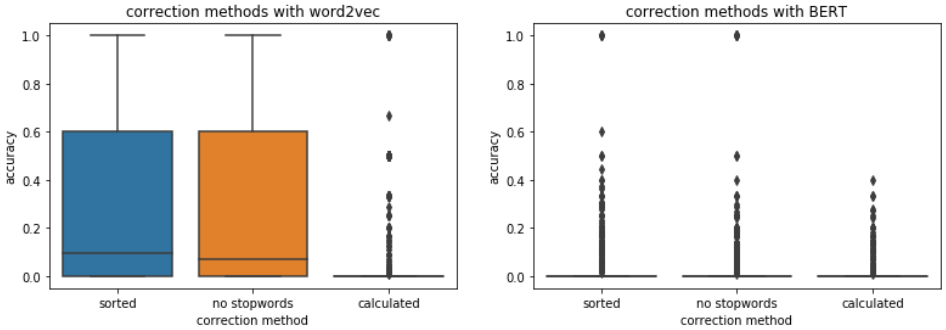


Figure 11: Performance of correction methods on the validation set. This figure compares the effectiveness of different ranking methods used for selecting correction candidates, highlighting the best performing approach for both word2vec and BERTje