Norbu Ketaka: Auto-Correcting BDRC's E-Text Corpora Using Natural Language Processing and Computer Vision Methods*

Queenie Luo[1]

Leonard W.J. van der Kuijp[2]
Harvard University

This paper discusses the Norbu Ketaka project, which employs the latest cutting-edge technologies in deep learning to process one million pages of Tibetan texts. This includes labeling training data, adjustments to neural-network architectures, and creating new Natural Language Processing (NLP) and Computer Vision models. Additionally, a staff administration system was designed and implemented using the Google Docs and Google Drive APIs, which facilitated the distribution of 12,000 documents to 40 part-time annotators based on their preferences and time availability, as well as tracking and recording the edited documents. We have donated one million pages of "cleaned" texts, along with the models and algorithms utilized in this project, to BDRC as the ***Norbu Ketaka*** collection which is publicly accessible on BDRC's website.[1]

1. *Introduction*

Locating exact words in ancient manuscripts is a crucial but time-consuming task for scholars in the humanities to study cultural, historical, and linguistic changes. In recent years, Optical Character Recognition (OCR) technology has greatly facilitated the progress made in the digitization of ancient

* Queenie Luo & Leonard W.J. van der Kuijp, "Norbu Ketaka: Auto-Correcting BDRC's E-Text Corpora Using Natural Language Processing and Computer Vision Methods", *Revue d'Etudes Tibétaines*, no. 72, Juillet 2024, pp. 26-42.

Listen to this article here: https://github.com/queenieluo/NorbuKetaka. We thank the Harvard China Fund for its generous sponsorship of this project. We are grateful to the Buddhist Digital Resource Center [BDRC] for their ready cooperation in providing text images and OCR-ed files, and the Department of East Asian Languages and Civilizations of Harvard University for extending additional administrative and financial support. Last but not least, we also thank Professor Kurt Keutzer of UC Berkeley for his generous support and comments on the technical aspect of this project.

1 Norbu Ketaka Collection: https://library.bdrc.io/show/bdr:PR1ER1?uilang=en

and more recent texts. The Buddhist Digital Resource Center (BDRC) [2] has archived over 15 million pages of culturally significant Buddhist works. Sponsored by Google's Tibetan OCR, BDRC digitized over 8,000 volumes of Tibetan Buddhist texts to create a searchable database. However, the digitized e-text corpus contains many errors that scholars cannot reliably use for their research. Some hand-written Tibetan graphs, such as "ཀ" and "ཏ", "ཇ" and "ཅ", are often indistinguishable even to the naked eye, and an OCR program cannot be expected to capture these nuances. Scholars in this field have to rely on their linguistic and philological knowledge to discern the correct errors based on word context and familiarity with the language. In addition, in the case of some of the texts' antiquity, most wooden block prints and manuscripts have faded ink and minor stains on pages that further challenge Google OCR's performance.

The original raw OCR output from Google's Tibetan OCR contains lots of "noise" or spelling errors due to misidentified characters in the image. For example, Google's Tibetan OCR system tended to read stains and black border lines as actual texts and produce several lines of unreadable textual noise. Current spelling-checkers cannot correct Tibetan texts, and are not designed to auto-correct OCR-ed output. The Norbu Ketaka project used the latest cutting-edge technologies in deep learning to clean one million pages of Tibetan texts, including labeling training data, tweaking neural-network architectures, and training models from scratch. We also implemented a staff administration system using Google Docs and Google Drive APIs to automatically dispatch 12,000 documents to 40 part-time employees based on their preferences and time availability, as well as track and record the edited document.

Fig. 1 shows the pipeline of this project. The raw images underwent an initial pre-processing stage using computer vision models, involving three distinct stages of rotation, border removal, and contrast adjustment, to minimize OCR errors. The images were then sent to Google OCR. Using the output from Google OCR, the system extracted each syllable's confidence score, which indicated the degree of accuracy in syllable identification. Subsequently, the system used these confidence scores to label both the images and the texts. These labeled images and texts were then inserted into a Google Docs document. A Tibetan BERT model was connected to the generated Google Docs documents to auto-correct low-level errors. The pre-cleaned documents were then dispatched to the annotators at Sichuan University via Google Drive. Once the annotators completed their corrections and uploaded the documents back to Google Drive, a staff administration program was activated to track

2 BDRC's official website: https://www.bdrc.io/

and record the returned documents against the dispatched documents to identify any missing documents. We then proofread the returned documents. Once all the documents were validated and approved, the texts were inserted into our database.

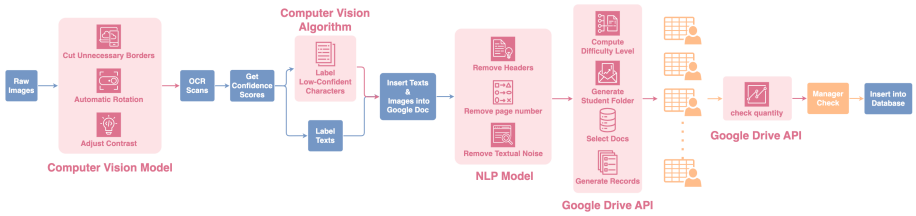

Fig. 1 – Workflow for processing images and texts, and distributing 12,000 documents to annotators.

2. *Literature Review*

The fields of computer vision and Natural Language Processing (NLP) have seen remarkable advances in recent years. In computer vision, several models have achieved notable prominence due to their remarkable performance in image recognition tasks. Convolutional neural-networks (CNNs) (LeCun, Bottou, Bengio & Haffner 1998) form the bedrock of many computer vision solutions. They are particularly effective due to their ability to process raw pixel data and learn complex patterns from images (Krizhevsky, Sutskever & Hinton 2017). Microsoft introduced ResNet (Residual neural-network), an advanced architecture built upon CNN (He, Zhang, Ren & Sun 2016). ResNet stands out for its remarkable capability to train exceptionally deep networks by incorporating "skip connections" or "shortcuts" that allow information to bypass certain layers. This approach effectively addresses the vanishing gradient problem, which can hinder training in deep networks. In addition to ResNet, other notable architectures include Mask R-CNN (He, Gkioxari, Dollár & Girshick 2017) and Imagenet (Krizhevsky et al. 2017). More recently, Facebook AI Research has developed Detectron2 (Wu, Kirillov, Massa, Lo & Girshick 2019), an advanced library that offers cutting-edge algorithms for image detection and segmentation. It supports various computer vision research projects and production applications in Facebook. In our project, we utilized the benefits of this publicly available Detectron2 model for image preprocessing. However, since Detectron2 is pre-trained on large and general datasets, it may not be perfectly adapted to domain-specific tasks. We fine-tuned this model by training it further on a smaller and task-specific dataset.

Parallelly, various Natural Language Processing (NLP) models have demonstrated robust performance in various language tasks, including spelling-error correction. The Transformer model, first introduced by Vaswani et al. in "Attention is All You Need" (Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser & Polosukhin 2017), has become a cornerstone in NLP. Its architecture includes attention mechanisms that allow the model to capture contextual relationships between words and thus solve the problem of long-term dependencies for long sentences. This has led to more sophisticated Transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers) (Devlin, Chang, Lee & Toutanova 2018). Developed by Google, BERT is pre-trained on a large corpus of text and has been proven effective in a variety of NLP tasks. It uses a masked language model to understand word context in both directions (left and right of a word). Since the primary objective of our project is to identify and eliminate low-level errors in OCR-ed texts, such as noise. The architecture of the BERT model has been proven to be particularly effective in this regard. To leverage this efficiency, we pre-trained a BERT Tibetan model and augmented it with an additional layer to perform a binary classification task. This setup allowed us to automatically identify incorrect or noisy text sequences from Google OCR's output.

Efforts have been made to utilize computational methods for processing Tibetan texts. Meelen, Roux & Hill (2021) showcase a pipeline capable of converting Tibetan documents in plain text or XML format into a fully segmented and part-of-speech (POS) tagged corpus. Implemented on the extensive collection from the Buddhist Digital Resource Center, this study shows the rule-based, memory-based, and neural-network methods led to an end-to-end accuracy rate of 91.99% for the automatic pipeline, making the resultant corpus a valuable resource for linguists and Tibetan studies scholars. Notably, other works have also explored tasks such as Classical Tibetan word segmentation and POS tagging (e.g., Jiangdi (2003); Kang, Jiang & Long (2013); Hill & Meelen (2017), Li, Li, Wang, Lv, Li & Duo (2022), Xiangxiu, Qun, Renqing, Nyima & Zhao (2022)). An & Long (2021) address the challenges of Tibetan dependency analysis by introducing a new dataset and proposing a neural-based framework. The proposed model achieves promising performance in Tibetan dependency analysis tasks by automatically extracting feature vectors for words and predicting their head words and dependency arc types. Our project does not focus on NLP tasks of word segmentation, POS tagging, and extracting feature vectors, but aims to clean and provide high-quality Tibetan dataset that supports research, analysis, and further advancements in NLP and the broader field of Tibetan studies.

3. *Technological Advancements in pre-processing data and generating documents*

This section covers the technological advancements employed in the Norbu Ketaka project. These advancements encompass pre-processing scanned images through computer vision neural-network models, utilizing the confidence scores from Google OCR output to label OCR-ed texts and images, employing the Google Docs API to generate and manage 12,000 Google Doc documents, and training and implementing a Tibetan BERT model to automatically correct low-level errors.

3.1 Pre-processing scanned images using computer vision neural-network models

The image pre-processing step involved three distinct stages: rotation, border removal, and contrast adjustment. The raw images used in the project were scanned copies of manuscripts or books, resulting in some images not being perfectly oriented with the vertical axis of the page. To tackle this issue, we developed a CNN rotation model specifically designed to rotate tilted images. The model architecture consisted of two convolutional layers, two max-pooling layers, and two fully connected layers. For training data generation, we selected 200 perfectly aligned images and employed PyTorch's torchvision library[3] to randomly rotate these images and recorded the rotation angles. The rotated images served as the inputs for the model, and the *negative* values of the rotation angles served as both the training labels and the model's outputs. To evaluate the accuracy of the rotation model, we utilized the Mean Squared Error (MSE) metric, measuring the discrepancy between predicted rotation angles and ground truth angles. The model achieved a low MSE, ranging between 0-0.5 degrees, after training for 100 epochs.

Next, we fine-tuned Facebook Research's Detectron2 (Wu et al. 2019) for the purpose of removing unnecessary borders in the images. Cutting the borders was crucial due to the significant number of errors and noise they introduced to the OCR output, particularly in the case of tainted images. For fine-tuning Detectron2, we utilized only 150 training labels. An example of the training data can be seen in Fig. 2, where the target bounding box was highlighted in red against a transparent background. These 150 training labels were hand-labeled. This model obtained 99.1% accuracy after 800 epochs of training. The errors were generally attributed to images that had fainted inks.

3 The torchvision.transforms.functional module in PyTorch's torchvision library provides a set of functional image transformation operations for data augmentation and manipulation, which can be accessed here: https://pytorch.org/vision/stable/transforms.html

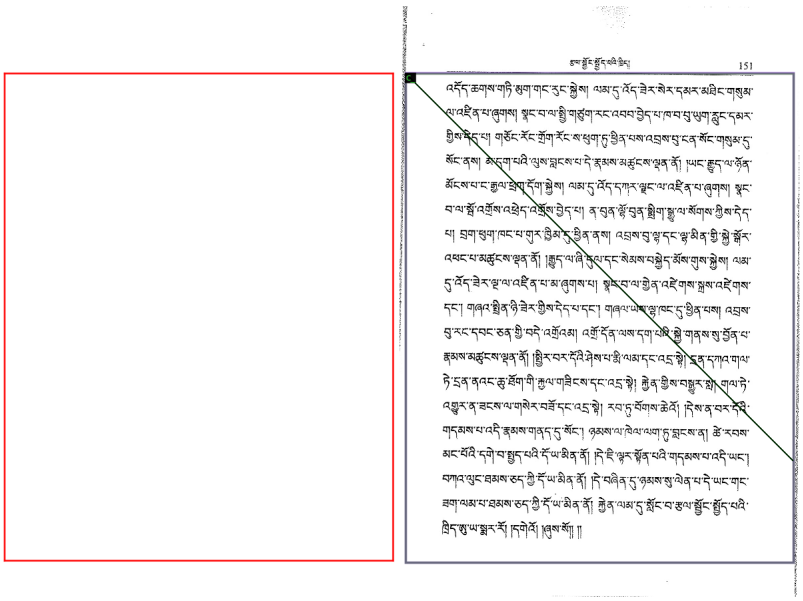

Fig. 2 – Training label for fine-tuning Detectron2: The red box on the left represents the actual training label, while the image overlaid with a bounding box on the right is for illustrative purposes.

Lastly, before sending them to Google's OCR system, we applied further contrast adjustments to the cropped images using the torchvision library.

Overall, the amount of OCR errors dramatically decreased using the pre-processed images by a factor of 2 - 10 (depending on the quality of the original image), compared to the amount of OCR errors using the original scanned images. Fig. 3 and Fig. 4 show comparisons between the original and pre-processed images.

3.2 Using the confidence score from Google OCR output to label OCR-ed texts and images

Next, we forwarded the pre-processed images to the Google OCR system. The OCR system's output included a confidence score for each identified character. Based on human validation, characters with confidence scores exceeding 80% exhibited nearly perfect accuracy, while characters with lower confidence scores, around 30% for example, often required correction. Thus, we utilized these scores as valuable indicators to label low-confident characters

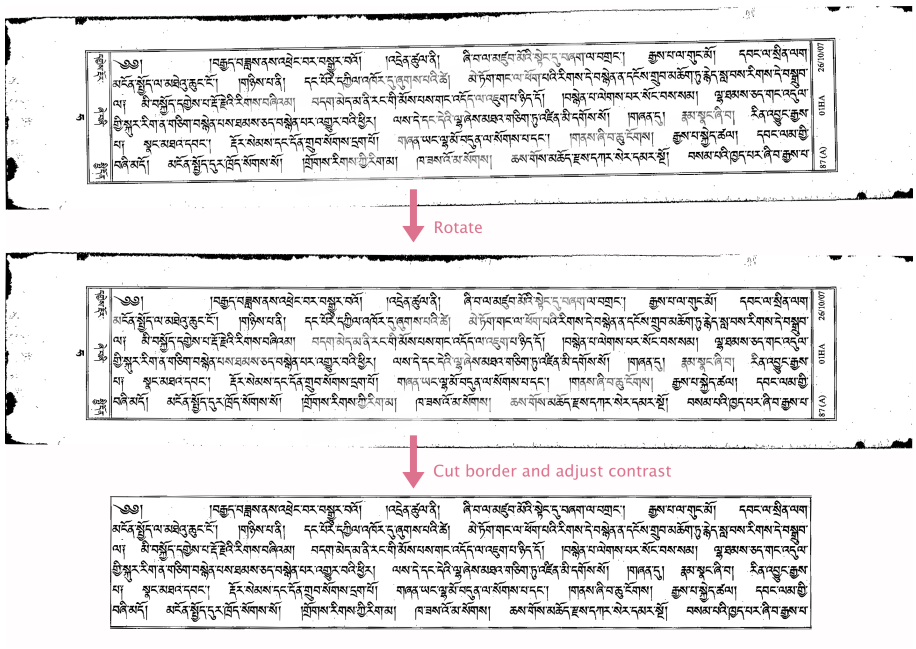

Fig. 3 – Comparison of original and pre-processed Images. Pre-processed images lead to a 2-10 fold decrease in OCR-ed errors.

and informed human annotators to focus on these characters (see Fig. 6).

However, it is important to note that the 80% confidence score threshold is not always an optimal choice for all texts. The quality of the texts could vary greatly, and some texts might have characteristics that made them more difficult for OCR to recognize accurately. For example, texts with brownish paper, red ink, or faded characters produced lower confidence scores even though the OCR output was correct (see Fig. 5). Additionally, Google OCR might have faced difficulties in identifying texts that contained a mixture of multiple languages. This was particularly true for bilingual dictionaries where Google OCR was unable to preserve the original two-column layout and accurately recognized Chinese characters. To solve these problems, we manually filtered out some texts and adjusted the confidence score thresholds for atypical texts.

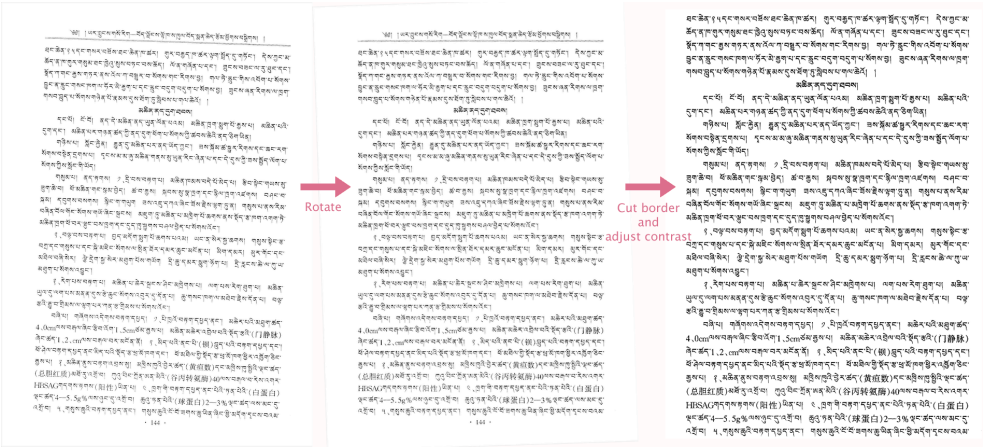

Fig. 4 – Comparison of original and pre-processed images. Pre-processed images lead to a 2-10 fold decrease in OCR-ed errors.

3.3 Using Google Docs API to generate and manage 12,000 Google Doc documents

To process one million pages of Tibetan texts, we had to generate around 12,000 Word documents, with each document containing a maximum of 100 pages of text. We set the maximum page number as 100 to avoid the document size growing too big. Managing such a large number of documents manually is not feasible. Thus, we utilized the Google Docs API[4] which allows for programmatic access and manipulation of Google Docs. By leveraging this API, we significantly improved the efficiency and accuracy of creating and managing large-scale data.

The Google Docs API allowed us to automate the creation and editing of Google Docs documents. We were able to seamlessly insert images and their respective OCR-processed texts into a single document, situating the image above its corresponding text (see Fig. 6). This resolved the issue of managing disjointed sets of images and texts. We also formatted low-confident characters in red to facilitate annotators to solely on the highlighted text within the documents, and edited the document's background color to light green to comfort their eyes.

4 Google Docs API is an interface provided by Google that allows developers to programmatically access and manipulate Google Docs documents, enabling automated document creation, editing, and collaboration: https://developers.google.com/docs/api.

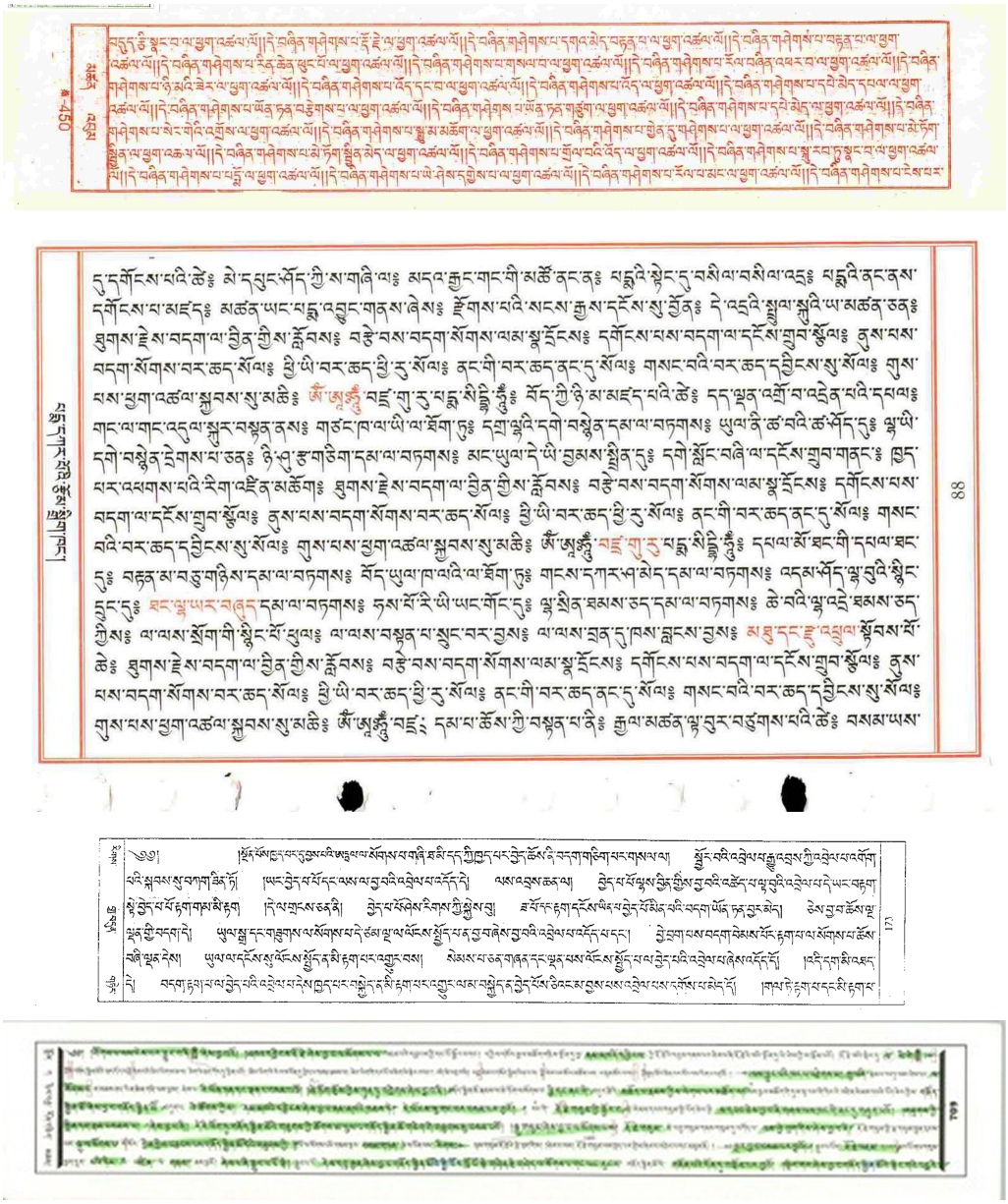

Fig. 5 – Sample images that tend to produce low confidence scores.

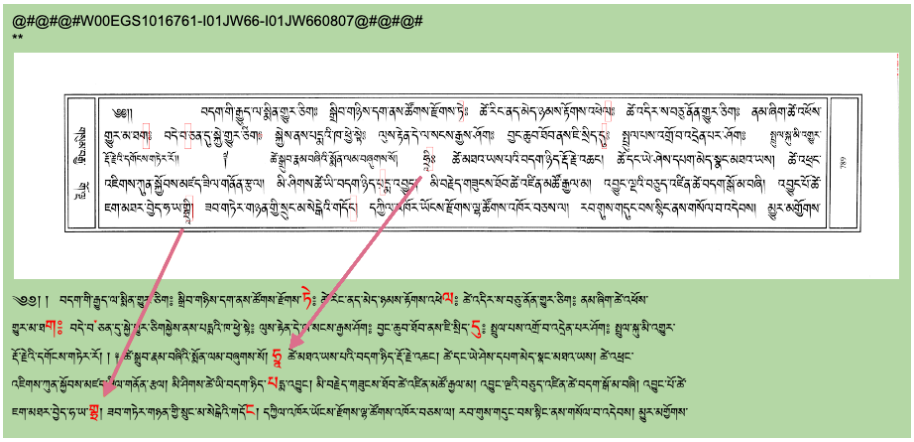

Fig. 6 – Screenshot of a Google document we generated using Google Docs API, in which the low-confidence characters are labeled in both the images and texts.

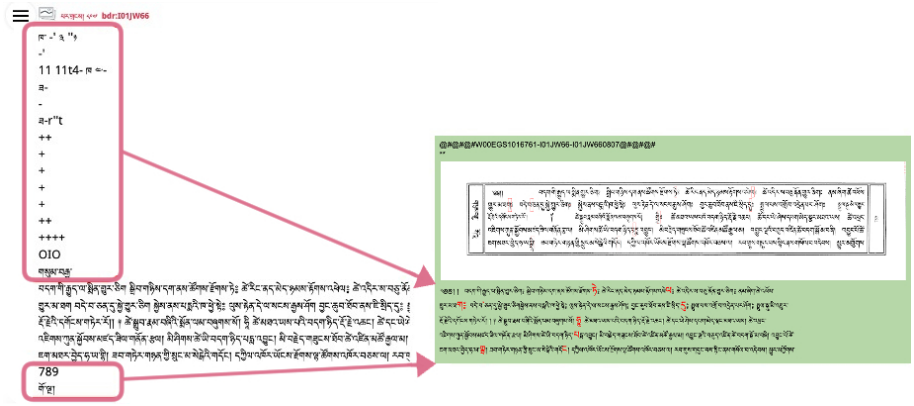

Fig. 7 – A comparison between BDRC's current website and our pre-processed texts. The image on the left is a screenshot from BDRC's website (acquired on 06/22/2022), and the image on the right is a screenshot of a Google document, in which all texts are pre-processed using our AI models.

Furthermore, we used Google Docs API to integrate the 12,000 generated documents with Google Drive and Google Cloud Storage[5] through Google Colab[6]

5 Google Cloud Storage is a scalable and durable object storage service provided by Google Cloud Platform, offering secure and flexible storage for various types of data with high availability and global accessibility: https://cloud.google.com/storage.

6 Google Colab is a cloud-based Jupyter notebook environment provided by Google that allows

and Google Drive API[7] to ensure every single document is ordered, counted, and proofread.

3.4 Training and using a Tibetan BERT model to auto-correct low-level errors

As mentioned above, the pre-processed images were able to significantly reduce noise and errors in the OCR output. However, we observed remaining errors that could be further edited using language models, such as page numbers, headers, and low-level spelling errors like repeated instances of "˝" and phrases that are mixed with non-Tibetan scripts. Therefore, we decided to train a Tibetan language model to auto-correct these low-level errors. It is important to note that in order to avoid the accidental correction of spelling variants, our language model does not intend to correct high-level mistakes in the main texts.

As introduced in the literature review section, the BERT architecture (Devlin et al. 2018) has been proven to be particularly robust in spelling error correction. Therefore, we chose the BERT architecture to implement a Tibetan spelling error correction model. We first pre-trained the BERT model using an architecture with 6 hidden layers, 12 attention heads, a maximum sequence length of 256 in which the tokenizer uses the byte pair encoding tokenizer algorithm to obtain subword units. The final tokenizer has a vocabulary size of 8,000. The training data consisted of 103 volumes of the Derge Kangyur and 213 volumes of the Derge Tengyur that are publicly accessible from the GitHub repository (Esukhia 2023). We then attached a fully connected layer to the pre-trained model to perform a binary classification task, wherein label 0 indicated that the given sentence was a standard Tibetan sentence, and label 1 indicated that the sentence was noise, a page number, or a header. Table. 1 presents examples of the two classes we wish to classify.

Fig. 8 shows the ROC curves of our Tibetan BERT model, which indicate the true positive rates versus the false positive rates for thresholds ranging from 0 to 1. BERT_fcc1, BERT_fcc2, and BERT_fcc3 represent three different fully connected layer configurations, with 64, 128, and 256 neurons, respectively. As indicated by the curves, the fully connected layer with 64 neurons presents the most favorable ROC curve, and we therefore adopted this model for our project. In our scenario, a false positive suggests the model erroneously classifying a correct sentence as a noisy one, leading to an un-

users to write, run, and collaborate on Python code seamlessly, with access to free GPU and TPU resources: https://colab.research.google.com.

7 Google Drive API is a programming interface provided by Google that enables developers to integrate their applications with Google Drive, allowing for file management, sharing, and synchronization: https://developers.google.com/drive/api.

Text	Label
བཅུ་གཅིག	1
Ir꤬ཆྃrཀྱིཟིཎ	1
ལ114rat	1
ཇཡཀདཆཀྱིFཀwwr	1
བའི་གནས་སུ་ཀུར་ཡ་ཞིག་ཏུ་སྣང་ངོ་།།	0
ཞལ་ནས་ཀྱང་རྨི་ལྟས་འདི་དག་ལ་དཔག་ན་ཙང་ལོ་ངེ་གཅིག་དང་""གཉིས་ཀྱི་ཐོག་ནད་ཚབ་ཅན་པོ་རྒྱུན་རིང་བྱང་བ་རིམ་གྱི་ ུ ེ ོ	0
དང་སྲས་ཅི་བ་གཉིས་གསེགས།བཙུན་མོ་དང་སྲས་གཞན་རྣམས་མ་མཐུན་པས་མངའ་འབངས་བགོ་ལ་འཁའ་བྱས་ཏེ་སོ་སོར་གྱེས་ ུ ེ ོ	0
རིད་ཀྱང་ལུག་སྤྱི་ལོ་གཉིས་ཀར་ན་ཚ་ཐུ་མོ་བྱུང་སོང་བས་ཡོན་མཆོད་གཉིས་ཀར་གེགས་ཆགা་ཅི་བའི་ལྟས་སུ་མངོན།།	0

Table 1 – Label 0 indicates the given sentence is a normal Tibetan sentence, and label 1 indicates the given sentence is either noise, page number or header.

intended deletion of an accurate sentence in the document; this forces annotators to manually retype the sentence. On the other hand, a false negative occurs when the model misclassifies a noisy sentence as a correct one, leading to the retention of a noisy sentence in the document; this requires annotators to manually remove it. In essence, erroneously deleting a correct sentence (a false positive) imposes a greater cost than failing to eliminate a noisy sentence (a false negative). Therefore, we set a high threshold of 0.8 for the model to maintain a low false positive rate. As can be observed from the ROC curve, with a threshold of 0.8, the false positive rate is close to 0.

After training and testing the model, we integrated it with our generated Google Docs documents via APIs to automatically identify and remove erroneous sentences. Prior to the automatic cleaning implemented by BERT, our annotators spent, on average, 20 minutes on the removal of these sentences in each document. With the integration of BERT, our annotators could now focus solely on the main texts.

The integration of computer vision models for image pre-processing and a Tibetan BERT model to address low-level errors led to an average reduction of errors in OCR output by 80% - 90% across various collections of Tibetan texts, significantly reducing labor and costs. In the next section, we discuss the challenges and solutions related to staff administration that we encountered during this project.

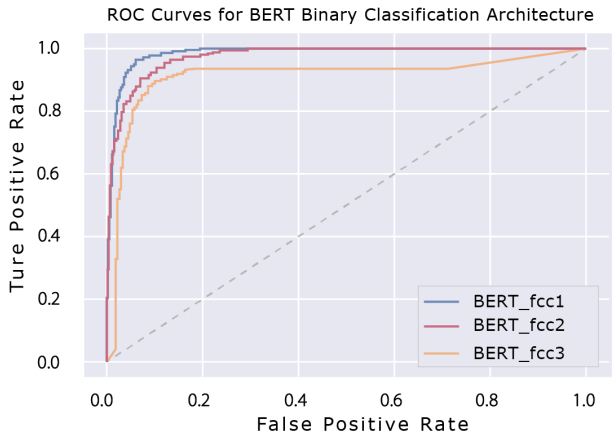

Fig. 8 – ROC curves for BERT binary classification architecture.

4. *Staff Administration*

Due to the vast number of documents and a large number of part-time employees, this project was not only a technical AI project. It was also a social-personnel management project. That is, we not only needed to ensure the high quality and accuracy of the models but also had to bear responsibility for recruiting and supervising the annotators at Sichuan University.

Our team was composed of 40 part-time employees. The generated documents presented varying amounts of errors - some demanded significantly more time to process than others, and each employee had different preferences and time availability. As a result, randomly distributing documents was impractical. To establish a fair and manageable workflow, we classified our documents into different difficulty levels and associated each level with a unit price. The difficulty levels were determined based on the count of low-confidence characters in the document. Table. 2 displays the estimated completion time corresponding to different document levels. Generally, level-0 documents took about 10 minutes to finish, while level-7 documents took 45 to 60 minutes. Employees were then assigned documents according to their preferred difficulty level.

Next, distributing the appropriate number of documents that meet employees' preferences is a complex mathematical problem. To address this issue, we implemented a system that utilized the Google Drive API to automatically dispatch, track, and record documents. Similar to the Google Docs API,

Level	Estimated Time	Number of Low-Confident Characters
0	10 - 13 mins	0 - 500
1	13 - 15 mins	500 - 1000
2	13 - 15 mins	1000 - 1500
3	15 - 17 mins	1500 - 2000
4	17 - 20 mins	2000 - 2500
5	20 - 30 mins	2500 - 3000
6	30 - 45 mins	3000 - 3500
7	45 - 60 mins	3500 - 4000

Table 2 – Estimated time required for different document levels. Level-0 documents typically take approximately 10 minutes to complete, whereas level-7 documents may require 45-60 minutes to complete.

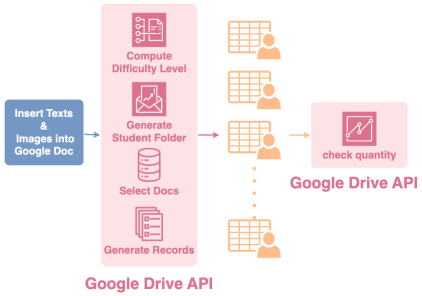

Fig. 9 – Staff administration pipeline to dispatch, track and check 12,000 documents.

the Google Drive API is capable of handling large quantities of files, making it ideal for organizing extensive document collections. Based on each employee's requirements, our system was able to automatically create a folder using the employee's name, calculate and select documents that align with the employee's preference, copy the relevant documents to the employee's folder, and generate an Excel sheet that records all the document names, their difficulty levels, unit prices, and the total payment for the entire folder (see Fig. 9).

Once the employees had finished editing and returned their documents, the system checked the dispatched documents against the returned documents to identify any missing documents. As depicted in Fig. 10, blue represents completed folders, while red represents incomplete ones. As indi-

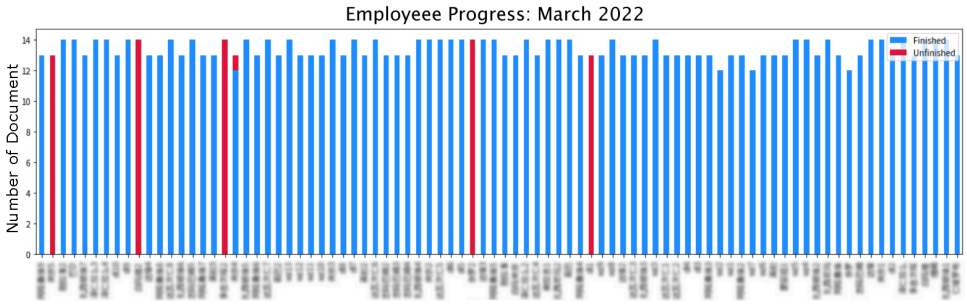

Fig. 10 – Status of employees' progress: The graph showcases the status of document completion, where blue present the completed documents and red denotes incomplete ones.

cated by the graph, one employee inadvertently omitted a document from their folder. In short, our staff administration system guarantees a seamless process for accurately distributing and collecting 12,000 documents, ensuring equitable compensation, personalized document selection based on employee preferences, and no missing documents.

5. Conclusion

In conclusion, we implemented two systems for this project: a technical system that integrated NLP and computer vision models to pre-clean Tibetan texts, and a staff administration system designed to dispatch, track, and record documents. We have donated one million pages of "cleaned" texts, along with the models and algorithms utilized in this project, to BDRC as the ***Norbu Ketaka*** collection. Ultimately, our project paves the way for linguistics and humanities research, offering a promising methodology for constructing extensive textual corpora in other under-researched languages. The innovative approach of our project can serve as a model for similar endeavors in processing old manuscripts on a large scale. The success of our project undoubtedly benefits scholars in the field of Tibetan studies, as well as the broader communities involved in NLP, and low-resource language research.

Bibliography

An, Bo & Congjun Long. 2021. Neural dependency parser for tibetan sentences. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 20(2). doi:10.1145/3429456. https://doi.org/10.1145/3429456.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee & Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)*, Association for Computational Linguistics.

Esukhia. 2023. Derge tengyur. https://github.com/Esukhia/derge-tengyur.

He, Kaiming, Georgia Gkioxari, Piotr Dollár & Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the ieee international conference on computer vision*, 2961–2969.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren & Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 770–778.

Hill, Nathan W & Marieke Meelen. 2017. Segmenting and pos tagging classical tibetan using a memory-based tagger. *Himalayan Linguistics* 16(2). 64–86.

Jiangdi, Kang Caijun. 2003. The methods of lemmatization of bound case forms in modern tibetan [c]. In *Ieee international conference on natural language processing and knowledge engineering*, .

Kang, Caijun, Di Jiang & Congjun Long. 2013. Tibetan word segmentation based on word-position tagging. In *2013 international conference on asian language processing*, 239–242. IEEE.

Krizhevsky, Alex, Ilya Sutskever & Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Communications of the ACM* 60(6). 84–90.

LeCun, Yann, Léon Bottou, Yoshua Bengio & Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11). 2278–2324.

Li, Yan, Xiaomin Li, Yiru Wang, Hui Lv, Fenfang Li & La Duo. 2022. Character-based joint word segmentation and part-of-speech tagging for tibetan based on deep learning. *Transactions on Asian and Low-Resource Language Information Processing* 21(5). 1–15.

Meelen, Marieke, Élie Roux & Nathan Hill. 2021. Optimisation of the largest annotated tibetan corpus combining rule-based, memory-based, and deep-learning methods. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 20(1). doi:10.1145/3409488. https://doi.org/10.1145/3409488.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones,

Aidan N Gomez, Łukasz Kaiser & Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems 30*, 6000–6010.

Wu, Yuxin, Alexander Kirillov, Francisco Massa, Wan-Yen Lo & Ross Girshick. 2019. Detectron2. https://github.com/facebookresearch/detectron2.

Xiangxiu, Cairang, Nuo Qun, Nuobu Renqing, Trashi Nyima & Qijun Zhao. 2022. Research on tibetan part-of-speech tagging based on transformer. In *2022 3rd international conference on pattern recognition and machine learning (prml)*, 315–320. IEEE.