A Method for the Calculation of Parallel Passages for Buddhist Chinese Sources Based on Million-scale Nearest Neighbor Search

Sebastian Nehrdich

Abstract

For research on Buddhist textual material, citations and similar passages are of major importance. This paper explores the application of continuous word representations and nearest neighbor search in order to efficiently compute a network of parallel passages for the Chinese Buddhist canon. It also discusses methods of evaluating the quality of the detected parallels and demonstrates a potential use case for the resulting data in the form of a web application for philological research.

1 Introduction

Much like Sanskrit and Tibetan literature, Chinese Buddhist literature has a high degree of internal cross-referentiality. Textual reuse and citations occur with high frequency, but the quoted sources are in many cases not explicitly mentioned and citations not necessarily marked as such. Such instances of textual reuse can be very literal, but paraphrases, change of grammatical structure, and the use of different vocabulary can also occur. Since the size of the canon of Chinese Buddhist literature is vast, digital methods for the detection of approximate parallel passages can be of great help to researchers.

This paper makes the following contributions to this problem:

• Two novel datasets for the evaluation of algorithms for the detection of parallel passages in Chinese Buddhist literature: one consisting of pairs of true parallel phrases and false parallel phrases that share close lexical features and the other consisting of phrase pairs which are different translations of the same non-Chinese source text (section 3).

• A pipeline for the detection of parallel passages for the entire CBETA collection based on phrase representations of fixed size and approximate nearest neighbor search (section 5).
A Method for the Calculation of Parallel Passages for Buddhist Chinese Sources Based on Million-scale Nearest Neighbor Search

- A web application that makes the generated data accessible in a convenient way and enables the user to search, compare and filter possible parallel passages on a large scale (section 7).

To the best of my knowledge, this is the first attempt to calculate approximate parallel passages for the entire CBETA collection and also the first attempt to solve this problem by the application of continuous word representations and approximate nearest neighbor search. The proposed method is largely language-agnostic and can be adapted to various scenarios, being especially suited for large corpora. The dataset and code are available at https://github.com/sebastian-nehrdich/buddhist-chinese-parallels. The web application is accessible at https://buddhanexus.net.

2 Definition

I kept the definition for what constitutes a relevant parallel intentionally rather loose in this study. I treated any two passages that share enough lexical and syntactical features, and that are as long as possible, as a relevant parallel passage.\(^1\) In the case of Classical Chinese,\(^2\) certain linguistic problems that are present in other Buddhist source languages do not play a role, while some special problems arise. Stemming is not necessary since morphological information is not encoded in the written language. But the question of the segmentation of phrases into semantic units is not trivial. Another very serious problem is the difficulty of determining sentence borders, since traditional Chinese writing prior to the twentieth century has generally been done in *scriptio continua*, without applying any kind of punctuation.\(^3\)

I use the following terminology in this paper: A *token* refers to a lexical unit that is

---

\(^1\) This definition is based on Hellwig (2013), but I added the criterion “shared syntactical features” since the co-occurrence of lexical units in close proximity alone is often not decisive enough to constitute a “relevant parallel” in Chinese Buddhist material.

\(^2\) In this study, I use the term “Classical Chinese” as an umbrella term for the language(s) of the Chinese literary tradition prior to the twentieth century, including translations of Buddhist texts, scholarly commentaries, poems, and other writings from both the Buddhist and the non-Buddhist tradition(s).

\(^3\) The situation is very similar in the case of Sanskrit, where South Asian manuscripts traditionally did not record word and sentence boundaries in the way that modern Western literature does.
considered relevant for the calculation of the parallels; stop words and punctuation marks are not considered to be tokens. A sequence is a series of tokens of a determined length. The term text refers to the different files as presented in the CBETA collection.

3 Dataset

3.1 Corpus

This paper uses the data contained in the Taishō Shinshū Daizōkyō (Taishō Tripitaka, 大正新脩大藏經, abbreviated “T”) and the Xuzangjing (卍續藏, abbreviated “X”). I obtained the data from the Chinese Buddhist Electronic Text Association (CBETA). These two collections together consist of 3,647 files, with a total size of about 450 MB, containing 460,756,903 characters including punctuation. Among the digitally available corpora on ancient Buddhist material, its size is rivaled only by the Tibetan corpus. The material contained in T and X shows a huge level of diversity in both content and language. The greater part of this material is autochthonous Chinese literature, while a considerable number of the texts in T are translations from various Central and South Asian source languages with the earliest examples dating back to the second century CE.

3.2 Datasets

In this paper, I introduce two new datasets for the evaluation of phrase similarity with regard to Chinese Buddhist literature.

4 The CBETA database is accessible at http://cbeta.org/. The raw data used as a basis for this study is accessible at https://github.com/cbeta-git/xml-p5a.

5 The combined size of the Tibetan Kangyur, Tengyur, and Sungbum collections as they are provided by the Asian Classics Input Project (https://asianclassics.org/) is about 580 MB. However, since the writing systems of the two languages encode the content in very different manners, this figure can be misleading. After compressing the files with the gzip algorithm, the size of T and X combined is 188 MB, while the Tibetan files together reach a size of 145 MB. The figure of the compressed files might be closer to the actual amount of information contained.
3.2.1 Phrase Similarity

One crucial task in the detection of approximately similar phrase pairs is to distinguish true, meaningfully similar phrase pairs from those that only have accidentally similar features but do not show meaningful similarity. While it is trivial to distinguish between identical phrase pairs and those that do not share much vocabulary with each other, the task becomes challenging when a certain degree of variation in vocabulary, word order, and grammatical structure within the approximately similar phrase pairs is allowed. I introduce a novel dataset for the evaluation of the performance of the phrase-similarity algorithms examined in this study. This dataset consists of 446 phrase pairs chosen from the entire T collection which meet the following criteria: Both phrases need to have at least one common subsequence of six tokens, which shares at least five identical tokens; the order of these tokens is allowed to vary. These criteria are required in order to exclude phrases with very different vocabulary or syntactical structure. These results have then been judged as being either “a case of borrowing/textual reuse” or “shares only similar lexical/grammatical features, but is not sufficiently related” by three independent annotators. The ratio of positive and negative samples in this dataset is equal. It needs to be emphasized that this classification is a subjective measure and can vary from annotator to annotator to some degree. Example 1 is a phrase pair from this dataset that displays meaningful similarity (the common subsequence of six tokens is underlined):

Example 1.

Phrase A: 如勝鬘經說。世尊。如來藏不空
Phrase B: 聖者勝鬘經言。世尊。不空如來藏。

Example 2 shows a pair that is only accidentally similar due to overlapping vocabulary, but does not constitute a meaningful parallel:

Example 2.

Phrase A: 如來十力所能成辦。由前所說如來十力
Phrase B: 今此如是無滅等十種句義，如前所說十種對治。

6 I am grateful to Marco Hummel and Junwei Song, who assisted me in this task.
3.2.2 Translation Similarity

In order to determine the performance of the models used in this paper on more distantly related phrases, 1,450 phrases of the two available Chinese translations of the Sanskrit text *Abhidharmakośabhāṣya* (Vasubandhu), one attributed to Paramārtha (真諦) (T.1559) and the other attributed to Xuanzang (玄奘) (T.1558), have been aligned with each other on the sentence level. In the same way, 1,450 phrase pairs have been taken from two different translations of the *Vimalakīrtinirdeśa*, one attributed to Zhi Qian (支謙) (T.474) and the other attributed to Kumārajīva (鸠摩羅什) (T.475). This dataset therefore has a total size of 2,900 phrase pairs.7

4 Previous Research

4.1 Parallel Passages in Related Corpora

Relevant attempts to calculate parallel passages for Buddhist and related material start with Prasad and Rao 2010, who calculated both exact and approximate parallels for a Sanskrit corpus based on string edit distance. They made use of the Smith-Waterman-Gotoh algorithm for local alignment for finding approximate parallels. In order to reduce the complexity, they divided the corpus into lines and sorted the lines by common prefixes.

Also based on string edit distance is the work of Klein et al. (2014), who used methods of dynamic programming to find all possible approximate parallel passages in the Tibetan canon. Shmidman, Koppel, and Porat (2018) demonstrated a method of efficiently finding parallel passages by representing each word by its two most infrequent letters, which reduces the computational complexity of finding potential parallel passages drastically.

Perhaps most relevant for the Chinese Buddhist canon is the work of Sturgeon (2018) on detecting textual reuse in pre-Qin and Han literature. In that study the problem of reducing the computational complexity was addressed by dividing the process into two stages: First, sets of identical strings8 with a length of four or more characters were calculated. Then a metric based on various factors such as similarity of the unnormalized version of the strings,

---

7 I am grateful to the Bibliotheca Polyglotta / Thesaurus Literaturae Buddhicae project for making these data available online: https://www2.hf.uio.no/polyglotta/index.php?page=library&bid=2.

8 The strings were normalized beforehand by removing the radical from compound characters.
lower weighting of grammatical particles, and the overall frequency of the characters over the entire corpus was calculated. In the second stage, this metric was then maximized over contiguous sequences of phrases in order to detect parallels which are longer than the minimum length. It needs to be noted that the corpus analyzed in Sturgeon’s study was much smaller than the corpus used in this study (less than 6 million words vs. more than 128 million tokens).

4.1.1 Semantic Similarity

In recent years continuous word representations have found wide application in a range of NLP tasks. Contrary to string similarity–based approaches, continuous word representations capture not only lexical but also semantic properties of tokens. Popular models for calculating word representations are those used by Mikolov et al. (2013), Pennington, Socher, and Manning (2014), and Bojanowski et al. (2017). Mikolov et al. (2013) demonstrated that unweighted averaging of word embeddings performs well for the representation of short phrases. Shen et al. (2018) have demonstrated a pooling strategy which can capture the spatial distribution of embeddings inexpensively. The main drawbacks of the previous string similarity–based approaches are their high computational cost and their inability to take into account the semantic similarity of the tokens. For a string-similarity algorithm, the difference between two tokens is always identical, no matter whether the two tokens are 君 and 水 or 二 and 三. For these reasons, I decided to use a combination of word representations and Hierarchical Navigable Small World graph (HNSW) indexing as a strategy to tackle these two issues in this study.

A phrase representation strategy that is suitable for this study has to meet two criteria: (1) calculation should be inexpensive enough to be accomplished for the entire dataset in reasonable time; (2) the size of the resulting representations should be small enough so that memory requirements for building an index do not become prohibitive.

Deep contextual embeddings such as those of Devlin et al. (2019) are therefore not

---

9 Approximate matching of inexact phrases based on string similarity is especially expensive. The problem of the prohibitive cost of calculating Levenshtein similarity for all possible phrase pairs even in the case of a relatively small corpus such as the 1.8-million-token Babylonian Talmud corpus was addressed by Shmidman, Koppel, and Porat (2018), who estimated the computing time for that calculation to be nearly twenty years.
applicable because of their high computational cost during both training and inference and their high memory footprint when applied on such a large dataset. Models which use neural architectures for the pooling of word2vec-type embedding are also disregarded for the same reasons. Based on the observation given by Shen et al. (2018), I assume that simple pooling strategies will give comparable performance, with the additional advantage of being cheap to obtain and apply.

5 Algorithm

Figure 1. Diagram of the entire pipeline for the generation of parallel passages.

The pipeline is demonstrated in figure 1. The individual steps are as follows:

5.1 Preprocessing

I disregarded all forms of punctuation since I assume that this will increase the quality of the results. I segmented the text material by character since single characters already carry a lot of semantic value in Classical Chinese. In Chinese Buddhist texts the use of longer compound words such as 阿賴耶識 (a phonetic rendering of the Sanskrit term ālayavijñāna) to denote
special technical terms, names, and concepts is common. To avoid indicating all occurrences of such compound words as possible parallel passages, I merged compound words with a length of four or more characters into single tokens based on the headword entries of the Digital Dictionary of Buddhism. After these preprocessing steps, the size of the canon is 128,191,170 tokens.

5.2 Word Representations

I use Fasttext (Bojanowski et al. 2017) as a word representation model because of its capacity to collect sub-word information, which can be helpful in the case of compound words. In order to address the problem of variant writings of characters, I restored the full texts in their different variants as recorded in the CBETA collection and trained Fasttext embedding with the skip-gram model and a dimensionality of 100 on this data. The resulting vocabulary has a size of 14,214 tokens.

5.3 Phrase Representation Method

I adapted three different pooling strategies (SWEM, Simple Word-Embedding-based Models) for the representation of short phrases from Shen et al. (2018): The average pooling method $SWEM^{avg}$, the maximum pooling method $SWEM^{max}$, and the hierarchical pooling method $SWEM^{hier}$. In $SWEM^{avg}$ the information of each vector element of the phrase is captured by addition. $SWEM^{max}$ only takes the maximum value along each dimension of the sequence of word vectors into account. Neither $SWEM^{avg}$ nor $SWEM^{max}$ can capture spatial information from the represented word vectors. For this reason, I also adapted the model $SWEM^{hier}$, which follows a hierarchical strategy. As a modification to $SWEM^{hier}$ as described by Shen et al. (2018), $SWEM^{avg}$ is called as the last step to obtain the single vector as a representation of the entire phrase instead of $SWEM^{max}$. For a detailed description of these different pooling methods, please see Shen et al. (2018). The phrase length in this study is six tokens, as a compromise to allow the detection of meaningful short phrases on the one hand.

---

10 The Digital Dictionary of Buddhism can be accessed at http://www.buddhism-dict.net/. My gratitude goes to Charles Muller, who gave me permission to use the data and helped me obtain the most recent version of the headword entries. For a recent publication on the Digital Dictionary of Buddhism, see Muller (2019).
hand and avoid the overgeneration of meaningless short results on the other hand.

5.4 Approximate Nearest Neighbor Search (ANN)

Sugawara, Kobayashi, and Iwasaki (2016) have shown that graph-based indexing has the best performance and precision for word representations. Therefore I used HNSW\textsuperscript{11} for indexing. Approximate nearest neighbor search (ANN) is always connected with lower-quality search results. As Boytsov et al. (2016) have demonstrated, the loss in precision when using HNSW ANN is just a few percent or even less and should not affect the usability of the generated data in a significant way.

6 Evaluation

In order to choose the best pooling strategy, I chose the following two evaluation strategies: (1) Accuracy/Precision/Recall/F1 scores on the dataset for phrase similarity and (2) Accuracy on nearest neighbor retrieval on the dataset for translation similarity. For the phrase similarity task, I added Levenshtein distance (LEV) as a baseline.

6.1 Phrase Similarity

For each subsequence with a length of six tokens in each phrase pair of the phrase similarity dataset, I calculated pooled vector representations. I chose the subsequence pair with a length of six tokens between the two phrases with the highest cosine similarity as representative for the similarity of the phrase pair, effectively disregarding the less similar parts of the sentence.

| Model        | A   | P   | R   | F1  |
|--------------|-----|-----|-----|-----|
| LEV          | 81.9| 83.0| 80.4| 81.6|
| SWEM\textsuperscript{avg} | 87.2| 89.2| \textbf{84.8}| 86.9|
| SWEM\textsuperscript{max}   | 87.4| 91.1| 83.0| 86.9|
| SWEM\textsuperscript{hier}  | \textbf{89.7}| \textbf{96.8}| 82.1| \textbf{88.9}|

\textsuperscript{11} The implementation is the Non-Metric Space Library (NMSLIB) (Malkov and Yashunin 2020), available at https://github.com/nmslib/nmslib because of its strong performance as described by Aumüller, Bernhardsson, and Faithfull (2018).
As the results show, LEV is already able to produce meaningful results in these tasks. This can be explained by the fact that LEV takes the order of the tokens into account, as meaningfully similar phrase pairs tend to reproduce the same set of tokens in a similar order. Among the vector-based approaches, SWEM\textsubscript{avg} and SWEM\textsubscript{max}, the two pooling methods which are not aware of the order of the tokens, outperform LEV by a considerable margin. This demonstrates that the ability of continuous word representations to capture semantic similarity of the individual tokens gives a considerable boost in performance as compared to string similarity–based methods. When compared with SWEM\textsubscript{avg} and SWEM\textsubscript{max}, SWEM\textsubscript{hier} shows the best accuracy with a higher precision score. This can be attributed to the fact that SWEM\textsubscript{hier} takes word order into account, which in the case of an isolating language such as Classical Chinese plays an important role. However, it is also noteworthy that both SWEM\textsubscript{avg} and SWEM\textsubscript{max} show higher recall scores than SWEM\textsubscript{hier}. This can be attributed to the fact that valid parallels can also sometimes have different word orders. The following example 3, which has been classified differently by the models, demonstrates the difference between taking the word-order into account and not doing so:

**Example 3.**

Phrase A: 是中亦無所成可得，故知異性無有詮表。

Phrase B: 如是果有所作，亦復不得，無異性故。

The passage relevant for the scoring is underlined. This pair has been classified by the annotators as not similar enough to constitute a parallel. In SWEM\textsubscript{hier} this is recognized as a true negative; in SWEM\textsubscript{avg} it is recognized as a false positive. SWEM\textsubscript{hier}, taking word order into account, shows a better ability to distinguish random parallels like these, which do not show much relationship with each other apart from the mere co-appearance of tokens, than SWEM\textsubscript{avg} or SWEM\textsubscript{max} do.

### 6.2 Translation Detection

For each possible subsequence with a fixed length \( L \) in phrase A of a phrase pair in the translation similarity dataset, I determined the closest neighbor among the possible phrases.

---

*Journal of the Japanese Association for Digital Humanities, vol. 5, No. 2, p. 141*
B of all of these phrase pairs. Since different translations of the same non-Chinese text vary more in vocabulary and style than the phrases of the previously evaluated phrase similarity, this is an interesting case for evaluating the ability of the models to capture more abstract semantic similarity. I repeated the experiment with a sequence length of $L = 6$ and $L = 10$ tokens. I calculated the accuracy in the following way:

$$A = \frac{\text{#correctly assigned sentences}}{\text{#all sentences}}.$$  

| Model   | $P$  |
|---------|------|
| $L = 6$ |      |
| SWEM$^\text{avg}$ | 30.8 |
| SWEM$^\text{max}$ | 29.1 |
| SWEM$^\text{hier}$ | 22.6 |
| $L = 10$ |      |
| SWEM$^\text{avg}$ | 44.5 |
| SWEM$^\text{max}$ | 39.2 |
| SWEM$^\text{hier}$ | 26.1 |

Table 2. Results for evaluation on translation similarity

As table 2 shows, the performance of SWEM$^\text{hier}$ is significantly worse with sequence lengths of either six or ten tokens than that of SWEM$^\text{avg}$ and SWEM$^\text{max}$. In the case of a sequence length of ten tokens, the difference between SWEM$^\text{avg}$ and SWEM$^\text{hier}$ is even greater than in the case of six tokens. I therefore conclude that for the detection of more distantly related Chinese Buddhist passages, disregarding the word order actually helps to improve the accuracy. Furthermore, the precision of bag-of-words-based models increases more with greater sequence lengths than is the case with the word-order-aware model. In order to understand how this is possible, I analyze an example from this experiment (example 4) that has been detected by the best-scoring model SWEM$^\text{avg}$ but not by the the word-order-aware SWEM$^\text{hier}$:

**Example 4.**

Paramārtha: 釋曰：即是有流思慧聞慧生得慧及助伴
Xuanzang: 慧謂得此有漏修慧思聞生得慧及隨行。
The underlined text represents the relevant characters for each phrase after stop-word removal. There is a significant overlap in the vocabulary of both phrases, but in the case of Xuanzang, the order of 思 and 慧 is reversed and 慧 is not repeated after 闻. Such changes do not have a great impact on the scoring in the case of a bag-of-words model like SWEMavg, but they make a big difference if word order is considered.

I repeated all experiments with the application of SIF weighting as described by Arora, Liang, and Ma (2017), but could not observe an improvement in performance. I also repeated all experiments after retraining the Fasttext representations with a dimensionality of 300, which did not produce notably different results.

### 7 Web Application

#### 7.1 Data Generation

For the web application, I calculated Fasttext representations with a dimensionality of 100 on the entire CBETA dataset with reconstructed variants. Based on the results of the evaluation, I decided that SWEMhier is the most appropriate pooling strategy for generating the dataset for the entire canon. While SWEMavg and SWEMmax show a better performance on detecting more distant paraphrases, their inferior performance on more literal parallels is problematic in view of the size of the canon and the huge number of possible parallel passages it contains. In test runs it became clear that SWEMavg and SWEMmax tend to overgenerate meaningless results, a problem that I did not encounter to the same degree with SWEMhier. I decided to use a sequence length of six tokens. I calculated the HNSW index using the default parameters of the NMSLIB implementation with the application of one postprocessing step. For each vector, \( k = 300 \) closest vectors have been retrieved and filtered based on their cosine distance. The maximum cosine distance for phrase pairs is 0.01, a

---

12 Sturgeon (2018) chose identical sequences with a length of four characters for the first stage of the algorithm. Here I set the minimum length to six tokens because the chosen pooling strategy allows for variation in vocabulary as well as word order.

13 The limitation of \( k = 300 \) was chosen in order to keep the memory requirements within reasonable boundaries. As a result, the precise detection of all possible parallel passages is no longer guaranteed if the
compromise between capturing enough possible parallels and excluding noisy, unrelated results.

I applied a couple of postprocessing steps to the results of the nearest neighbor search. Because of the allowed variance in cosine similarity, nonmatching characters are frequently included at the beginnings and ends of matches. Therefore I used local alignment\(^{14}\) on the detected parallels to guarantee an optimal detection of the boundaries of the matches. Like Klein et al. (2014) and Sturgeon (2018), I merged clusters of parallels in close proximity to form longer parallels. The maximum allowed distance for merging two subsequent sequences into a parallel is twelve tokens, counted from the first common token; therefore, taking overlap into account, the actual gap allowed for an unrelated insertion is six tokens. In this way the algorithm captures overlapping parallels, as well as parallels that contain a shorter diverging passage but begin and end with a parallel. As a constraint, the gaps between the embracing parallels in the target and in the source text have to be approximately of the same size, allowing a maximal variance of four tokens in both directions. After reconstructing the filtered stop words and punctuation, I removed parallels shorter than five characters, excluding punctuation. Finally, I restored the punctuation as present in the CBETA source files.

The entire process, including the generation of the Fasttext representations, the generation of the HNSW index, and the local alignment, takes about three days on 4x Xeon CPUs. The HNSW index for this dataset requires about 120 GB of memory.

7.2 Web Interface

This process generates a vast amount of data, and the number of recorded parallels for certain texts and passages can be overwhelming. For this reason, we\(^{15}\) designed a user interface with

\(^{14}\) I used the Smith–Waterman algorithm (Smith and Waterman, 1981). Its use for this purpose has already been described by Klein et al. (2014).

\(^{15}\) My gratitude at this point goes to Ven. Ayya Vimala, who contributed a large amount of their time and knowledge to designing and constructing this web interface. I would also like to thank Orna Almogi, who...
flexible display options and filters to limit the amount of data presented.

Figure 2. The website displaying T.1600 (辯中邊論) as the main text in text view mode.

Figure 3. T.1600 (辯中邊論) and T.1599 (中邊分別論) side by side in text view mode.

7.2.1 Text View Mode

supported me in many ways with her valuable feedback during the design of the web interface, and Michael Radich, who gave valuable feedback regarding the quality and presentation of the Chinese material.
In the text view mode, which is the default mode of the web interface, the main text is displayed on the left-hand side with each part that has a possible parallel being highlighted and clickable. The possible parallels are displayed in a column to the right of the main text in the middle of the window after the user clicks on a highlighted passage in the main text.

Very frequently, different parallels are layered on top of each other, sometimes forming clusters with hundreds of parallels in close vicinity. We use color coding to indicate how many parallels are present at a certain position of a text. Light blue indicates the presence of one parallel, deep blue the presence of two parallels; as the number increases, the coloring progresses through red toward yellow, which indicates the presence of ten or more parallels. We assume that ten or more quotes indicate a certain level of popularity of a parallel and that further progressive highlighting is not of much practical use. Figure 2 shows how the text view mode looks in practice. The main text is displayed on the left side; the column on the right side displays the possible parallels in descending order according to their normalized Levenshtein similarity. In figure 2, a full quotation of the verse present in T.1585 (成唯識論) is selected, which causes the corresponding section of the main text T.1600 to be highlighted with a gray background color. This strategy of coloring the different characters according to the number of occurring parallels results in a heat map that can give a quick impression about the location and quantity of the parallels contained within one text.\(^\text{16}\)

7.2.2 Dual Text View Mode

After clicking on any of the parallels in the right-hand column in the text view mode, a third column opens up on the right side to display the text that contains the parallel selected previously. This is illustrated in figure 3. In the right-hand column of figure 3, the currently selected parallel as well as other possible parallels that have been detected between the text in the left-hand column and the text in the right-hand column are highlighted. When clicking on a highlighted passage on the right-hand side, the corresponding parallel is displayed in the middle column and the main text on the left-hand side is automatically scrolled to the

\(^{16}\) This solution is similar to the solution presented by Sturgeon (2018), where different shades of red are used to indicate layers of quotations. However, unlike Sturgeon, we did not highlight the differences in wordings between two passages visually. This is because the data also contain examples of less verbatim quotations, where such a highlighting strategy becomes difficult to apply.
corresponding position. In this view mode it is also possible to disable the middle column to devote more screen space to the display of the two texts. This mode provides a synoptic overview of the parallels between two texts, enabling the user to compare two texts without the constant need to switch windows.

7.2.3 Table View Mode

In the table view mode (fig. 4) it is possible to display only the parallels without the surrounding text in a table format and then filter and sort them according to various criteria. It is possible to sort the parallels according to their position in the main text, grouped by the texts in which they appear and by their length.

![Figure 4. Screenshot of the website displaying only the parallels between T.1600 and T.1830 sorted by their length, beginning with the longest.](image)

7.2.4 Filter Options

In order to limit the number of displayed parallels, we offer various filter options:
1. String similarity, using normalized Levenshtein distance as a base for calculation.

2. Length, using the minimum length based on the length of the parallel excluding punctuation.

3. Number of co-occurrences, using the number of co-occurrences that a parallel has.\textsuperscript{17}

4. By text name and category, providing an option to filter by either including or excluding results from selected texts or categories.

Increasing or decreasing the minimum required string similarity has the effect of displaying more or less literal parallels. Changing the minimum required length removes or adds short parallels which might not necessarily show a strong relation to the analyzed text. Changing the number of allowed co-occurrences per parallel has a great effect when it comes to showing or hiding stock phrases such as “如是我聞：一時，佛 [...]” which might be of little interest to the user. Filtering by text names and categories is useful when one wants to compare a specific text against another text, set of texts, or category. Being able to put certain texts and categories on a “block list” is helpful to filter out quotes from certain sources that are not of interest to the user.

7.3 Discussion of the Results

An evaluation of the detected parallels over the entire size of the corpus is rather difficult since, to the best of my knowledge, no comprehensive reference data exist. Nevertheless, a few examples of the data I have generated should be discussed at this point. The strategy of down-weighting stop words makes the algorithm rather robust when it comes to detecting paraphrases in which the general word order has been kept, but certain function words have been inserted or skipped. Example 5, from T.1585 and its commentary T.1830 (成唯識論述記), demonstrates this clearly (the characters which are relevant for scoring are underlined):

\textsuperscript{17} This score represents how many times a parallel is contained within other parallels. For a parallel $A$, the co-occurrence score is increased by one for each other parallel $B_m$, $B_{m+x}$ that contains the parallel. As a criterion, $B_m$ needs to be at least of the same length (with a tolerance of two tokens) or longer than $A$ to affect the co-occurrence scoring of $A$. 

---

Journal of the Japanese Association for Digital Humanities, vol. 5, No. 2, p. 148
Example 5.

T.1585: 又無心位此類定無。既有間斷性非堅住。如何可執持種受熏。

T.1830: 位無心時此類定無。是識類故。如識間斷。性非堅住。故不可執持種受熏。

There can be little doubt about the fact that the commentator in T1830 is referring to the phrase in the text T1585 that he is commenting on. If the stop words and gaps of these two phrases had been considered during scoring, the result would have gotten a low ranking and consequently have been excluded from the list of possible results.

However, in the case of short phrases, false detection of results with little relevance can also become a problem. For example, the following result (example 6) detected between T.1599 (中邊分別論) and T.1600 (辯中邊論) demonstrates such a difficulty:

Example 6.

T.1599: 增益無損減。應知

T.1600: 增益及損減見。若知

While the vocabulary certainly hints at a semantic relationship between these two passages, the actual wordings and grammatical structures of the two phrases are different enough to make this at least a borderline case. The appearance of such results can be reduced in the web application by increasing the minimum required length and minimum required string similarity for a parallel to be displayed.

8 Conclusion

In this study I introduced a pipeline consisting of word representations, hierarchical pooling, and approximate nearest neighbor search for the detection of similar passages for the Chinese Buddhist canon and two datasets for the evaluation of the performance of algorithms to detect semantic similarity of Chinese Buddhist material. The required computation time of about three days on four Xeon CPUs for the generation of parallels within the entire CBETA collection is a tiny fraction of the time and computing power that a brute-force string similarity comparison of the sequences of a corpus of this size would require. The method described is largely language-agnostic. As long as meaningful word representations can be calculated, the pooling steps, the nearest neighbor search, and the postprocessing steps
remain the same. The pooling strategy can be adjusted to the language: This study has shown that hierarchical pooling performs well for Chinese Buddhist material, but in the case of a language such as Sanskrit, where word order is much less fixed, a different pooling strategy might produce better results.

The data that I generated in this study can be used in a variety of downstream applications. Here we demonstrated a web application that gives the user easy access to the texts and their detected possible parallels in the CBETA collection. Among possibilities for future work, the application of graph theory as described by Hellwig (2013) and Bingenheimer (2018) shows great potential for the creation of meaningful visualizations of these data.

Another very promising direction for further studies is the integration into this pipeline and web application of cross-lingual representation models, such as XLM, described by Conneau and Lample (2019), which can enable the detection of parallel passages across different pairs of languages.
References

Arora, Sanjeev, Yingyu Liang, and Tengyu Ma. 2017. “A Simple but Tough-to-Beat Baseline for Sentence Embeddings.” In 5th International Conference on Learning Representations (ICLR 2017): Conference Track Proceedings. N.p.: OpenReview.net. https://openreview.net/forum?id=SyK00v5xx.

Aumüller, Martin, Erik Bernhardsson, and Alexander Faithfull. 2018. “ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms.” In Similarity Search and Applications. SISAP 2017, Lecture Notes in Computer Science 10609, edited by Christian Beecks, Felix Borutta, Peer Kröger, and Thomas Seidl, 34–49. Cham: Springer. doi:10.1007/978-3-319-68474-1_3.

Bingenheimer, Marcus. 2018. “Who was ‘Central’ in the History of Chinese Buddhism?: A Social Network Approach.” International Journal of Buddhist Thought & Culture 28 (2): 45–67. doi:10.16893/IJBTC.2018.12.28.2.45.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. “Enriching Word Vectors with Subword Information.” In Transactions of the Association for Computational Linguistics 5, 135–46. https://transacl.org/index.php/tacl/article/view/999.

Boytsov, Leonid, David Novak, Yury Malkov, and Eric Nyberg. 2016. “Off the Beaten Path: Let’s Replace Term-Based Retrieval with k-NN Search.” In Proceedings of the 25th ACM International Conference on Information and Knowledge Management, 1099–1108. New York: ACM. doi:10.1145/2983323.2983815.

Conneau, Alexis, and Guillaume Lample. 2019. “Cross-lingual Language Model Pretraining.” In Advances in Neural Information Processing Systems 32 (NeurIPS 2019), 7059–7069. http://papers.nips.cc/paper/8928-cross-lingual-language-model-pretraining.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. “BERT: Pretraining of Deep Bidirectional Transformers for Language Understanding.” In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, Long and Short Papers, 4171–86. N.p.: Association for Computational Linguistics. doi:10.18653/v1/N19-1423.

Hellwig, Oliver. 2013. “Googling the Rishi: Graph Based Analysis of Parallel Passages in...
A Method for the Calculation of Parallel Passages for Buddhist Chinese Sources Based on Million-scale Nearest Neighbor Search

Computational Linguistics (Volume 1: Long Papers), 440–50. N.p.: Association for Computational Linguistics. doi:10.18653/v1/P18-1041.

Shmidman, Avi, Moshe Koppel, and Ely Porat. 2018. “Identification of Parallel Passages Across a Large Hebrew/Aramaic Corpus.” Journal of Data Mining and Digital Humanities, special issue on Computer-Aided Processing of Intertextuality in Ancient Languages. https://jdmdh.episciences.org/4175.

Smith, T. F., and M. S. Waterman. 1981. “Identification of Common Molecular Subsequences.” Journal of Molecular Biology 147 (1): 195–97. doi:10.1016/0022-2836(81)90087-5.

Sturgeon, Donald. 2018. “Unsupervised identification of text reuse in early Chinese literature.” Digital Scholarship in the Humanities 33 (3): 670–84. doi:10.1093/llc/fqx024.

Sugawara, Kohei, Hayato Kobayashi, and Masajiro Iwasaki. 2016. “On Approximately Searching for Similar Word Embeddings.” In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2265–75. N.p.: Association for Computational Linguistics. doi:10.18653/v1/P16-1214.