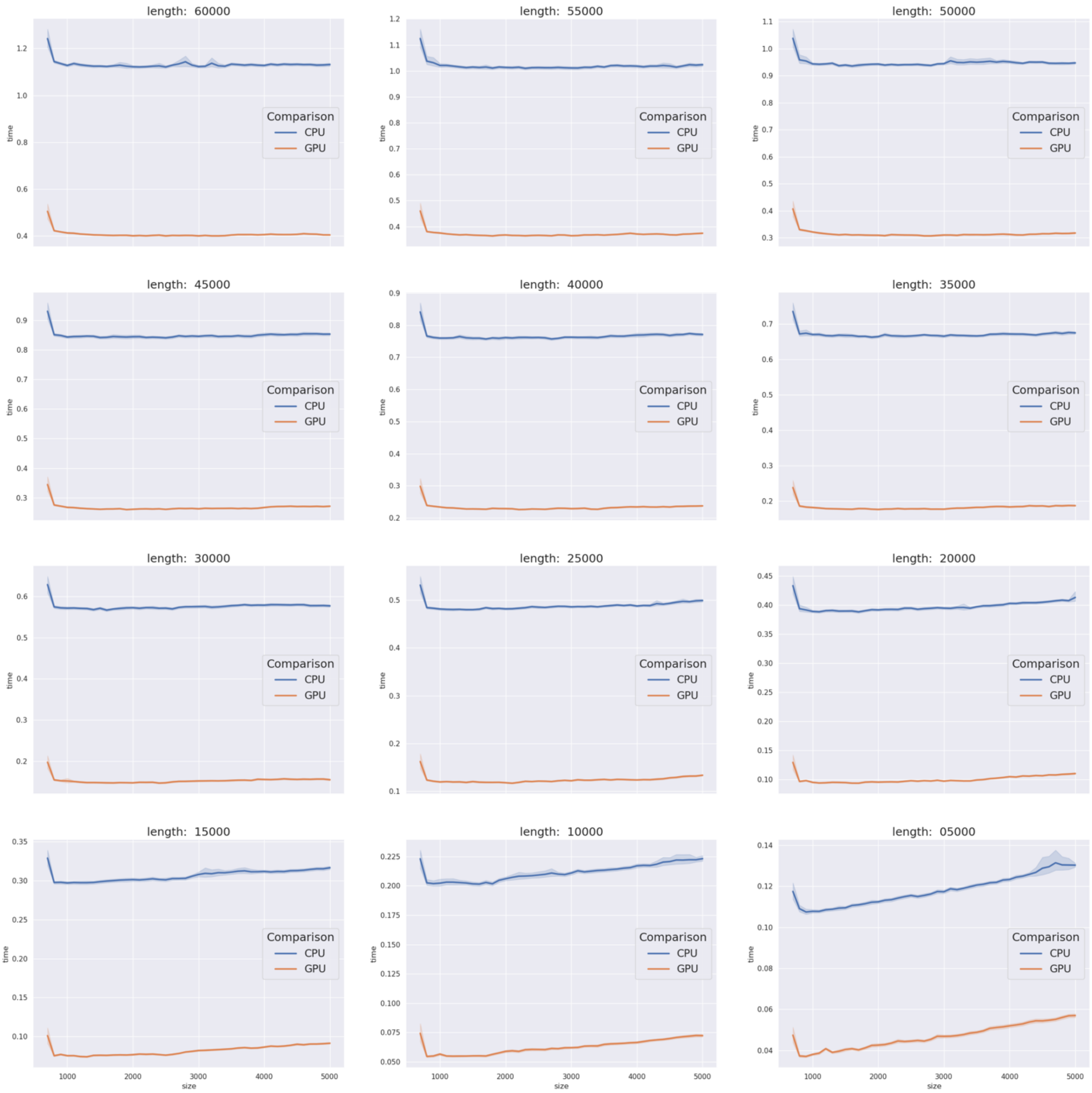
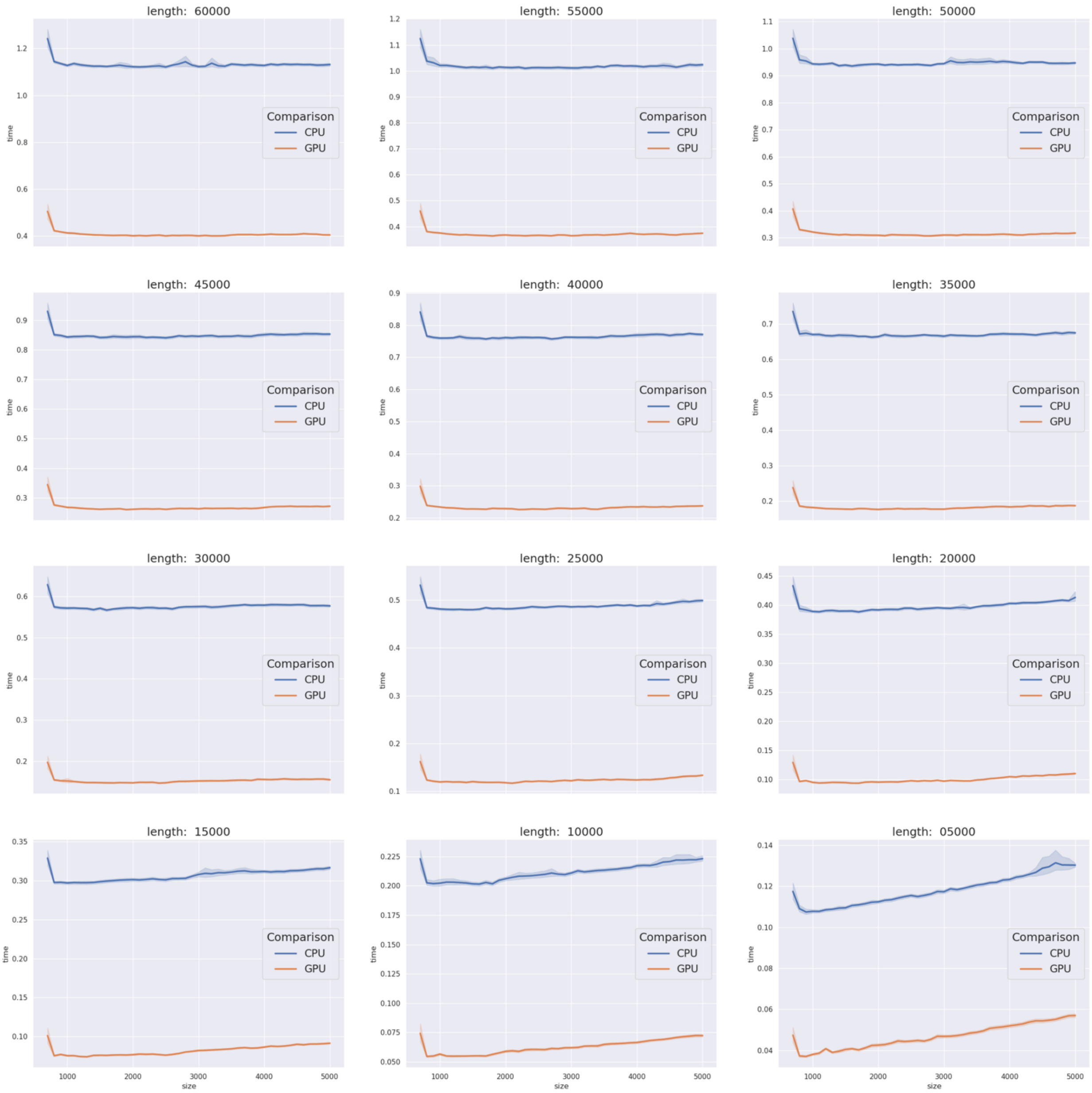
ПРИЛОЖЕНИЕ А

Графики зависимости времени работы от размера текста



ПРИЛОЖЕНИЕ Б

Графики зависимости времени работы от количества пакетов



ПРИЛОЖЕНИЕ Б

Графики зависимости времени работы от количества пакетов

```
def flexible_batch_indices(text, approximate_batch_size):
    """Find the borders of the batches.
    Using batches is much more efficient than raw text.

    Parameters
    ----------
    text : str
        Text for analysis.
    approximate_batch_size: int
        Estimated batches size in characters.

    Returns
    -------
    batch_indices: list
       List of indices of the end of batches.
    """
    exp = r’[.?!](?= [A-Z]|$)’
    cur_index = 0
    find_start = 0
    batch_indices = [0]
    sentence_found = False
    # Continue while start index < text length.
    while find_start < len(text):
        find_start = cur_index + approximate_batch_size
        find_end = find_start + approximate_batch_size

        # Check if the right index is less or equal to the length
           of the text.
        if find_end > len(text):
            find_end = len(text)

        match = re.search(exp, text[find_start:find_end])
        # If a match is found, recalculate the indices and add to
           the list.
        if match:
            cur_index = match.end() + find_start - 1
            batch_indices.append(cur_index)
            sentence_found = True
        else:
            # Just shift the index otherwise.
            cur_index += approximate_batch_size
            sentence_found = False
    if sentence_found:
        batch_indices.append(len(text))

    return batch_indices
```

```
def flexible_batch_indices(text, approximate_batch_size):
    exp = r’[.?!](?= [A-Z]|$)’
    cur_index = 0
    find_start = 0
    batch_indices = [0]
    sentence_found = False
    # Continue while start index < text length.
    while find_start < len(text):
        find_start = cur_index + approximate_batch_size
        find_end = find_start + approximate_batch_size

        # Check if the right index is less or equal to the length
            of the text.
        if find_end > len(text):
            find_end = len(text)

        match = re.search(exp, text[find_start:find_end])
        # If a match is found, recalculate the indices and add to
            the list.
        if match:
            cur_index = match.end() + find_start - 1
            batch_indices.append(cur_index)
            sentence_found = True
        else:
            # Just shift the index otherwise.
            cur_index += approximate_batch_size
            sentence_found = False
    if sentence_found:
        batch_indices.append(len(text))

    return batch_indices
```