Clean and preprocess the text

First, I need to read file through url by using request library

```python
# read data
url = "https://www.gutenberg.org/files/11/11-0.txt"
response = requests.get(url)

if response.status_code == 200:
    text = response.text
else:
    print("Failed to fetch the text:", response.status_code)
```

After I have built a function to normalize words in text. This function will remove punctuation, digits, space.

```python
def normalize(word):
    word = word.lower()
    word = re.sub(r'[^\w\s]', '', word)  # remove punctuation
    word = re.sub(r'\d+', '', word)      # remove digits
    return word.strip()
```

I remove stop words in text by using a pre-built set of wordcloud (will be updated by using Nltk library , and hope it will work better). Beside I add some stop words I found text.

```python
stopwords = set(STOPWORDS)
custom_stopwords = {'s', 'i', 'ii', 'iii', 'iv', 'v',
                    'vi', 'vi', 'vii', 'viii', 'ix', 'x', 'xi', 'xii', 'xiii', ''}
stopwords = stopwords.union(custom_stopwords)
```

Stop words set:

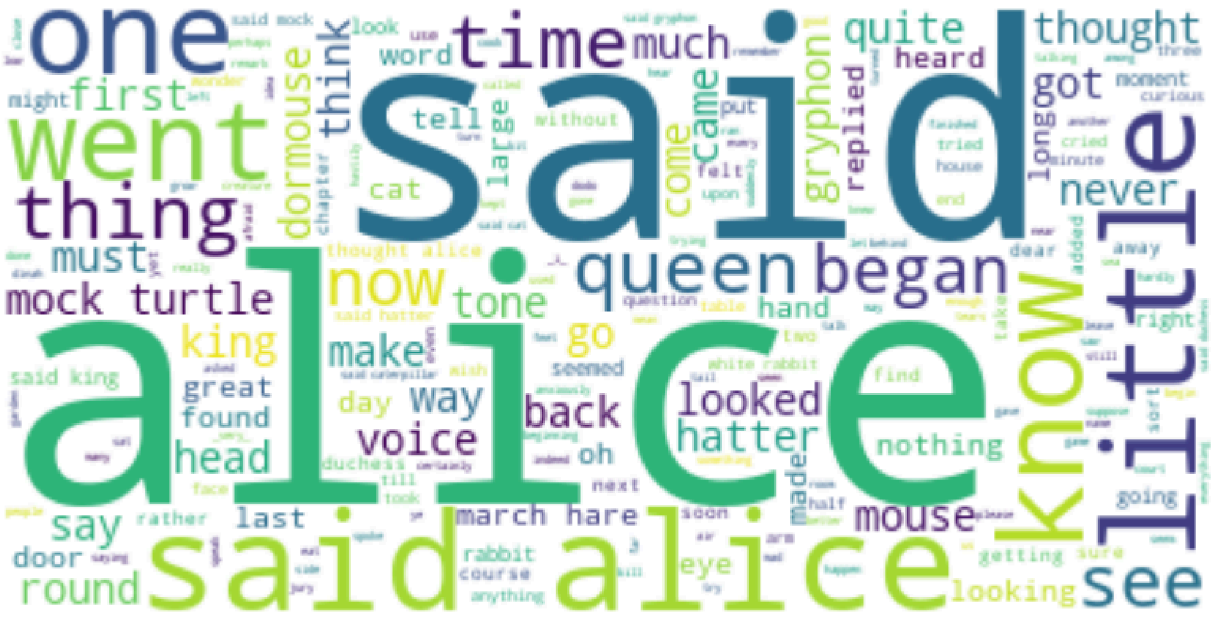
```
{'whens', 'ii', 'more', 'can', 'im', 'nor', 'here', 'hasnt', 'we', 'viii', 'your', 'down', 'do', 'ive', 'by', 'at',
'its', 'i', 'therefore', 'it', 'have', 'http', 'very', 'where', 'wouldnt', 'my', 'theyd', 'any', 'out', 'who', 'its
elf', 'well', 'theyve', 'so', 'vii', 'this', 'a', 'after', 'in', 'than', 'up', 'whats', 'isnt', 'while', 'vi', 'can
t', 'shant', 'hows', 'just', 'heres', 'and', 'theres', 'ourselves', 'but', 'with', 'not', 'few', 'those', 'agains
t', 'xi', 'own', 'him', 'such', 'x', 'having', 'wasnt', 'werent', 'yours', 'above', 'however', 'over', 'too', 'of',
'could', 'only', 'other', 'were', 'both', 'be', 'under', 'ours', 'shouldnt', 'as', 'to', 'there', 'would', 'whom',
'before', 'once', 'you', 'lets', 'being', 'shed', 'how', 'also', 'doesnt', 'shell', 'hadnt', 'was', 'shes', 'whys',
'id', 'these', 'their', 'some', 'herself', 'else', 'what', 'ill', 'get', 'xii', 'below', 'her', 'no', 'like', 'he
d', 'his', 'they', 'whos', 'k', 'because', 'for', 'them', 'ix', 'or', 'since', 'himself', 'youll', 's', 'he', 'didn
t', 'xiii', 'between', 'are', 'cannot', 'again', 'the', 'theyll', 'an', 'until', 'yourself', 'our', 'during', 'he
s', 'wed', 'youve', 'why', 'me', 'on', 'weve', 'she', 'all', 'couldnt', 'yourselves', 'iii', 'into', 'off', 'abou
t', 'ought', 'hence', 'has', 'had', 'further', 'which', 'am', 'iv', 'havent', 'otherwise', 'themselves', 'then', 'w
hen', 'www', 'youd', 'r', 'mustnt', 'doing', 'been', 'shall', 'if', 'same', 'theyre', 'youre', 'does', 'hell', 'tha
ts', 'arent', 'ever', 'that', 'through', 'is', 'hers', 'myself', 'most', 'each', 'should', 'theirs', 'v', 'com', 'f
rom', 'wheres', 'did', 'wont', 'dont'}
```

Before I normalize text then text has **26543** words
After I have normalized text then text has only **11971** words , reduce more ½ original words

Visualize word frequencies

- ### WordCloud

To quickly grasp the most frequent and important words in text



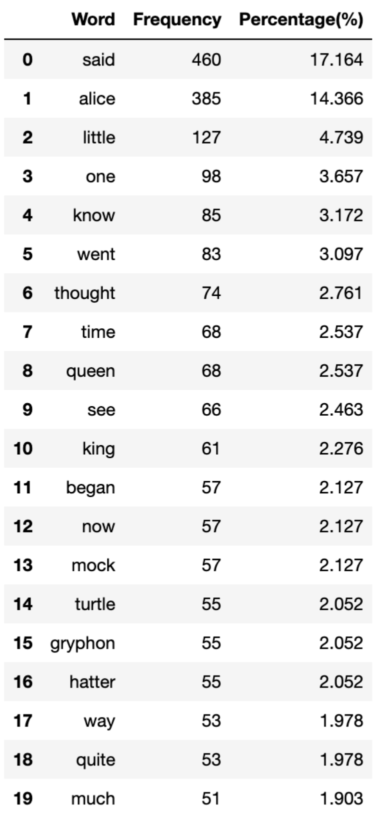
The text of Alice Wonderland so I can conclude that:

- **'said'** and **'alice'** are 2 words used at most
    - Dialogue heavily used in the story
    - Alice is main character
- **'little'**, **'one'** and **'thing'**, ...
    - They are descriptive storytelling
- **'think'**, **'went'**, **'got'** and **'come',** …
    - Character's movement and actions
- **'queen'**, **'king'**, **'hatter'**, **'tuttle'**, **'mouse'**, …
    - Other characters in this story

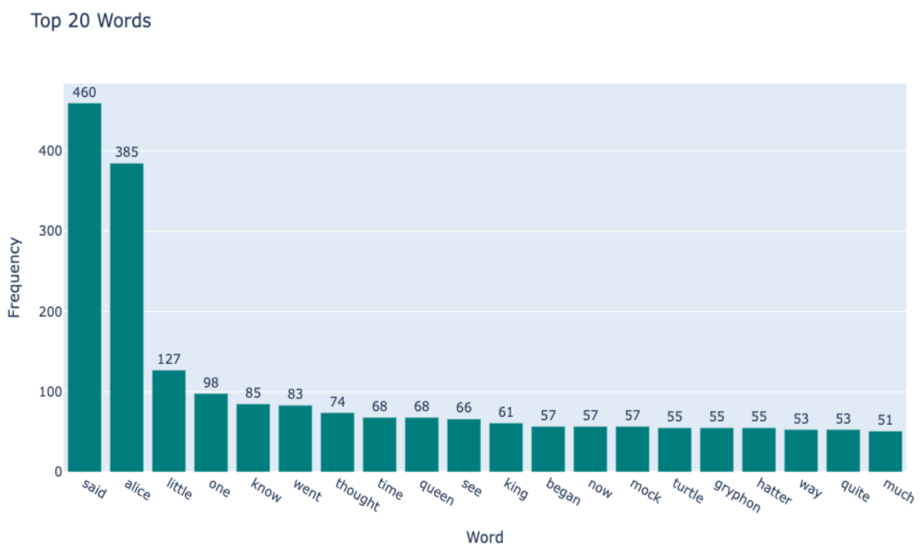
- ## Bar chart

I have used the Counter library to count word frequencies and calculate its percentage stored all into a dataframe. After that I used the Plotly library to plot bar chart.

Due to there are **11971** words so I only plot top **20** words to analysis



	Word	Frequency	Percentage(%)
0	said	460	17.164
1	alice	385	14.366
2	little	127	4.739
3	one	98	3.657
4	know	85	3.172
5	went	83	3.097
6	thought	74	2.761
7	time	68	2.537
8	queen	68	2.537
9	see	66	2.463
10	king	61	2.276
11	began	57	2.127
12	now	57	2.127
13	mock	57	2.127
14	turtle	55	2.052
15	gryphon	55	2.052
16	hatter	55	2.052
17	way	53	1.978
18	quite	53	1.978
19	much	51	1.903



Most Frequent Words

- "said" (17.16%) is the most frequently used word - **460** times.
- "alice" (14.37%) is the second most frequent - **385** times.

Beside Alice is a main character there are some secondary characters appearing in the story like king, queen, hatter, gryphon, mock, and turtle. In addition, there are action words like began, see, know, went show their interactions.

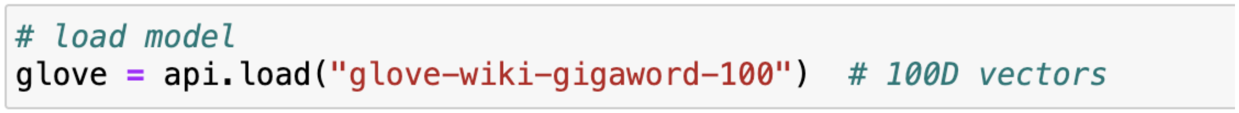
→ this is a dialogue , indicated by ‘said’ . Text is Alice in wonderland → Alice is main character

Analysis semantic word relationships

This part I will use GloVe embeddings and PCA

- GloVe embedding: vectorize word (word to vector) into a matrix (100 dimensional).
- PCA: reduce dimensional from **100** to **2** (more easier analysis).

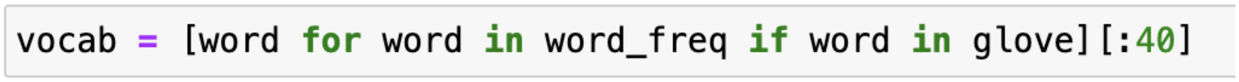
First, I need to load model Glove



```
# load model
glove = api.load("glove-wiki-gigaword-100")  # 100D vectors
```

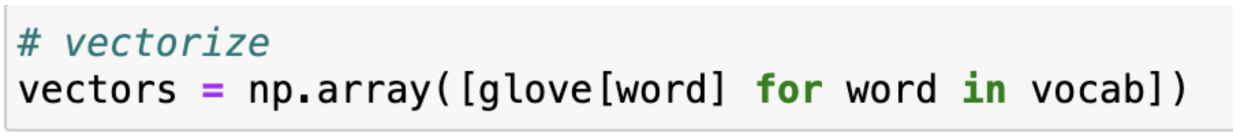
It takes a long time to load , so only load once.

Second, due to there are more 1000 words so I will extract 40 words to plot a heatmap (more easier analysis).



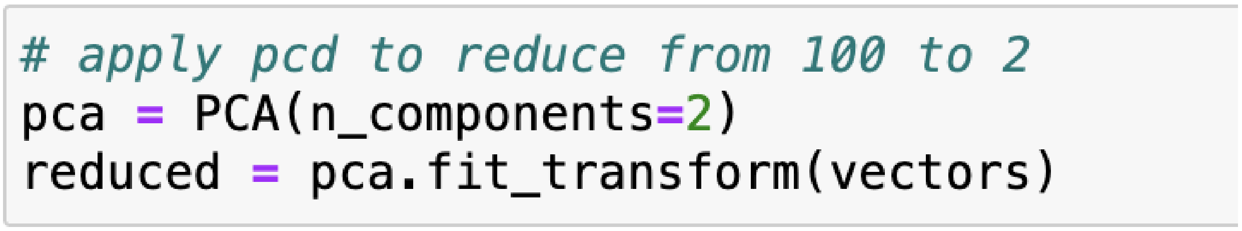
```
vocab = [word for word in word_freq if word in glove][:40]
```

After that I vectorize them.



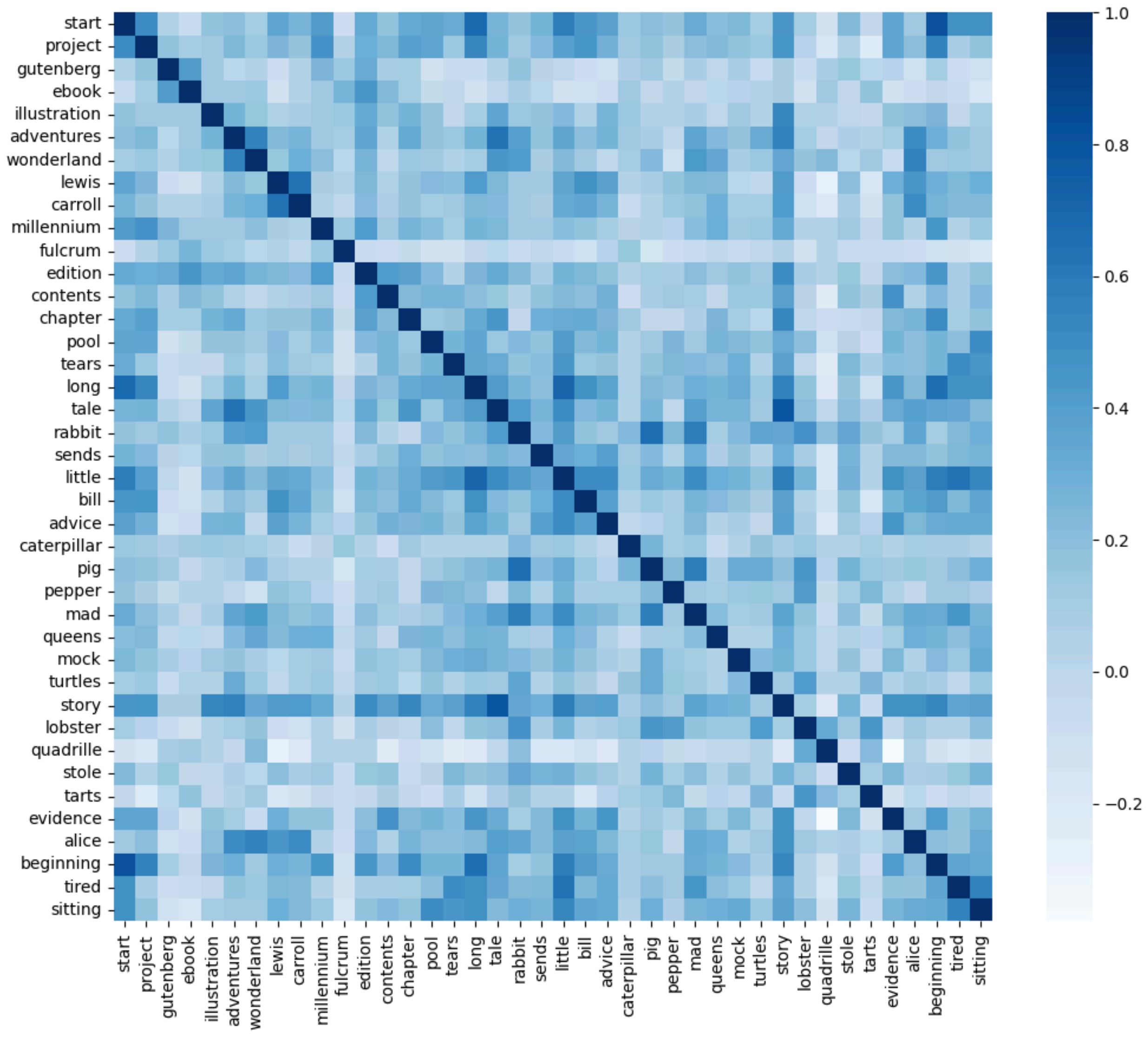
```
# vectorize
vectors = np.array([glove[word] for word in vocab])
```

Finally, I applied PCA to reduce the dimension.



```
# apply pcd to reduce from 100 to 2
pca = PCA(n_components=2)
reduced = pca.fit_transform(vectors)
```

Here's a heatmap plotted.

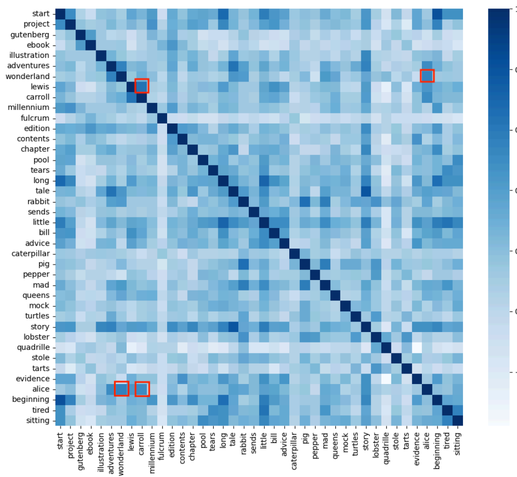


Note: I analyzed heatmap base Glove and it trained on emonous and wiki data.

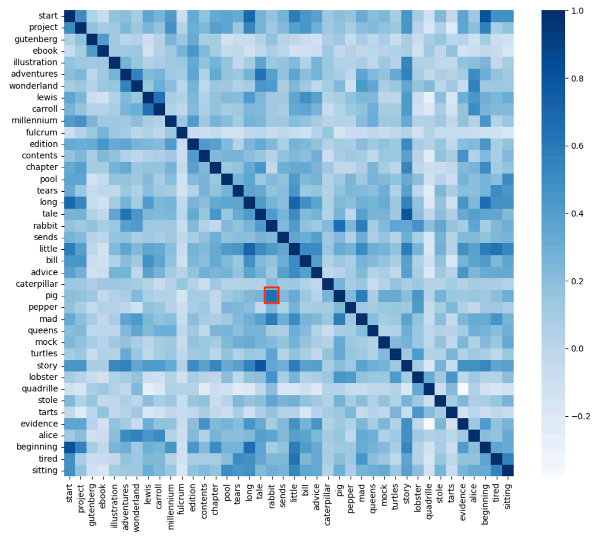
Dark diagonal line: each word is perfectly similar to itself (**1**)

Off-diagonal: darker/lighter areas indicate weaker or negative semantic similarity.

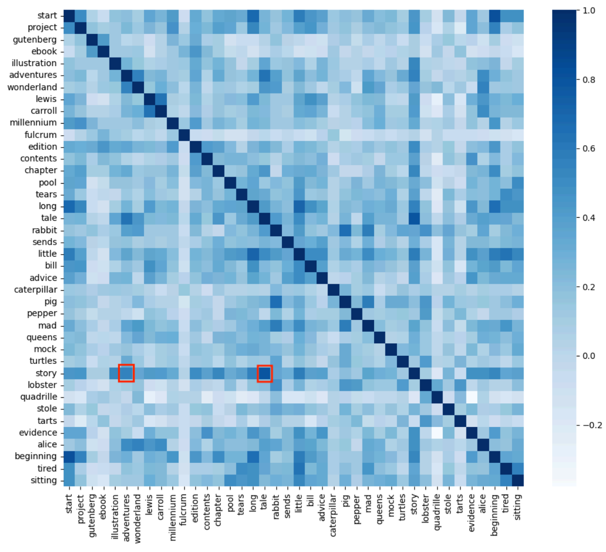
"**lewis**", "**carroll**", "**alice**", "**wonderland**" semantically related to the story: Alice main character in Alice in wonderland story and Lewis Carroll is author



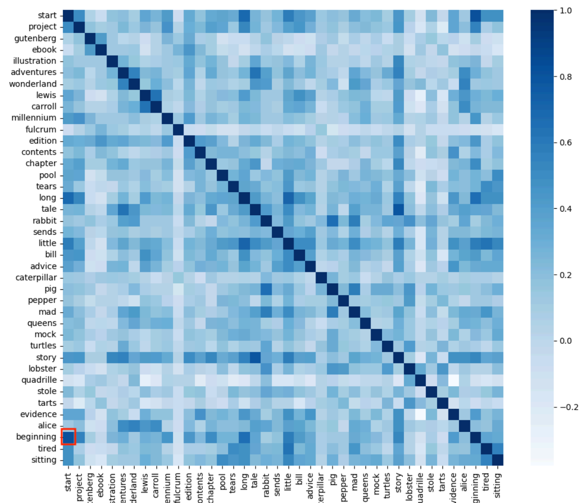
‘pig’ and **‘rabbit’** are characters/animals in *Alice in Wonderland*.



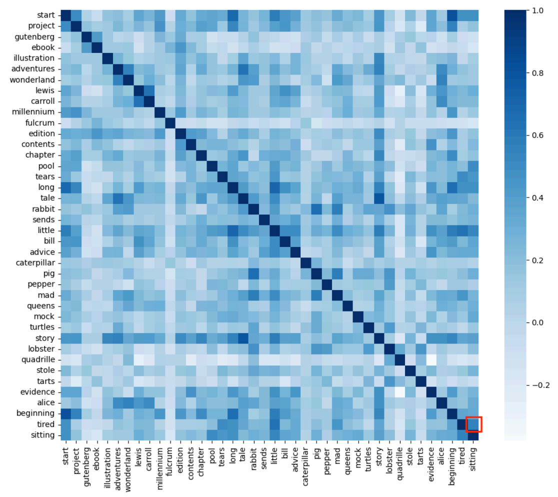
‘story’, **‘adventure’** , **‘tale’** we usually use tale and adventure to talk about type of story



‘beginning’ and **‘start’** are same meaning so of course they will be semantic similarity

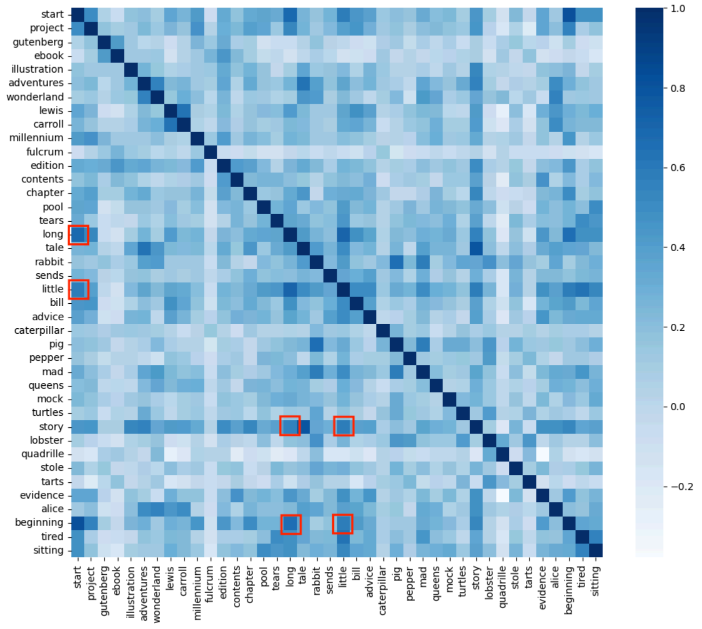


‘sitting’ and **‘tired’** are dark due to when you feel tired then you will sit



Finally, I remind again I am using GloVe to plot heatmap and there are areas that show strong relationships but I don’t know why. In my opinion it’s trained on emonous data , not only on Alice in wonderland story.

‘start’, **’beginning’**, **‘story’**, **‘little’** and **‘long’**



I checked the similarities between them.

```python
print(f'start and long: {glove.similarity("start", "long")}')
print(f'start and little: {glove.similarity("start", "little")}')

print(f'\nstory and long: {glove.similarity("story", "long")}')
print(f'story and little: {glove.similarity("story", "little")}')

print(f'\nbeginning and long: {glove.similarity("beginning", "long")}')
print(f'beginning and little: {glove.similarity("beginning", "little")}')
```

```
start and long: 0.6825861930847168
start and little: 0.5960179567337036

story and long: 0.5482589602470398
story and little: 0.5896822214126587

beginning and long: 0.6497789621353149
beginning and little: 0.5924113988876343
```