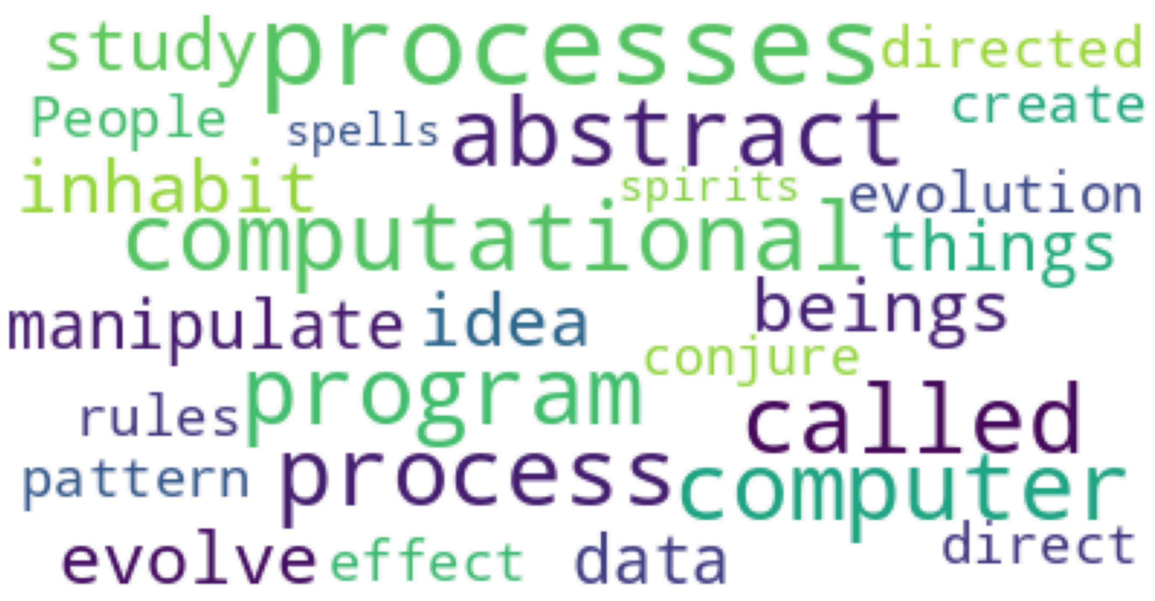
```
import re
import numpy as np
import string
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline

from subprocess import check_output
from wordcloud import WordCloud, STOPWORDS
```

```
stopwords = set(STOPWORDS)
sentences ="""We are about to study the idea of a computational process.
Computational processes are abstract beings that inhabit computers.
As they evolve, processes manipulate other abstract things called data.
The evolution of a process is directed by a pattern of rules
called a program. People create programs to direct processes. In effect,
we conjure the spirits of the computer with our spells."""
```

```
wordcloud = WordCloud(
                          background_color='white',
                          stopwords=stopwords,
                          max_words=200,
                          max_font_size=40,
                          random_state=42
                         ).generate(sentences)

fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(10, 10))
axes.imshow(wordcloud)
axes.axis('off')
fig.tight_layout()
```



```
sentences = """We are about to study the idea of a computational process.
Computational processes are abstract beings that inhabit computers.
As they evolve, processes manipulate other abstract things called data.
The evolution of a process is directed by a pattern of rules
called a program. People create programs to direct processes. In effect,
we conjure the spirits of the computer with our spells."""
```

```
# remove special characters
sentences = re.sub('[^A-Za-z0-9]+', ' ', sentences)

# remove 1 letter words
sentences = re.sub(r'(?:^| )\w(?:$| )', ' ', sentences).strip()

# lower all characters
sentences = sentences.lower()
print(sentences)
```

```
we are about to study the idea of computational process computational process
```

```
words = sentences.split()
vocab = set(words)
print(words)
print(vocab)
```

```
['we', 'are', 'about', 'to', 'study', 'the', 'idea', 'of', 'computational', '
{'direct', 'we', 'by', 'other', 'create', 'inhabit', 'conjure', 'that', 'with
```

```
vocab_size = len(vocab)
embed_dim = 10
context_size = 2
```

```
word_to_ix = {word: i for i, word in enumerate(vocab)}
ix_to_word = {i: word for i, word in enumerate(vocab)}
print(word_to_ix)
print(ix_to_word)
```

```
{'direct': 0, 'we': 1, 'by': 2, 'other': 3, 'create': 4, 'inhabit': 5, 'conju
{0: 'direct', 1: 'we', 2: 'by', 3: 'other', 4: 'create', 5: 'inhabit', 6: 'co
```

```
data = []
for i in range(2, len(words) - 2):
    context = [words[i - 2], words[i - 1], words[i + 1], words[i + 2]]
    target = words[i]
    data.append((context, target))
print(data[:5])
```

```
[(['we', 'are', 'to', 'study'], 'about'), (['are', 'about', 'study', 'the'],
```

```
embeddings =  np.random.random_sample((vocab_size, embed_dim))
print(embeddings)
```

```
[[4.10744134e-01 9.70470642e-01 7.21952784e-01 4.32854823e-01
  7.96969673e-01 4.04660540e-02 7.44067409e-01 3.77606926e-01
  2.29132802e-02 3.19514492e-01]
 [1.93346010e-02 7.45227876e-01 3.72021447e-01 7.73159287e-02
  2.62723461e-02 4.99686416e-01 2.51208269e-01 1.09833965e-02
  2.54451073e-02 6.90534486e-01]
 [1.85049218e-01 8.09941334e-01 6.92674008e-01 9.90107751e-01
  4.13597752e-01 1.87988423e-01 5.25218468e-01 6.87753381e-02
  9.11168736e-01 2.05274462e-01]
 [9.34875863e-01 1.48803634e-01 9.90609237e-01 1.45373713e-01
  4.52064194e-01 8.25736495e-01 5.39879892e-01 3.77581076e-01
  7.22230264e-01 1.30270628e-01]
 [6.99099706e-01 4.35451176e-01 9.59940131e-01 1.48657798e-01
  2.29092748e-01 7.57573451e-01 9.02636628e-03 7.98276358e-01
  5.32991804e-01 4.93954669e-01]
 [7.96076311e-01 6.82211111e-01 1.47407316e-02 1.72451604e-01
  5.28702937e-01 3.03369029e-01 7.85900284e-01 6.89807336e-01
  7.10074400e-01 6.02629802e-01]
 [8.13851555e-01 7.73503500e-02 3.34119044e-01 9.45791482e-01
  3.31848793e-01 4.53518473e-01 6.81405476e-01 3.95252444e-01
  3.60275442e-01 2.73057088e-01]
 [6.30232679e-01 9.27853907e-01 4.17114939e-01 9.32412935e-01
  3.02715492e-01 1.23138743e-01 3.89749808e-01 4.84469494e-01
  4.17726570e-01 3.98749070e-01]
 [6.42357295e-01 7.38876115e-02 2.32292683e-01 5.87955237e-01
```

```
  2.36628598e-01 7.38782777e-01 7.21453734e-01 1.81525933e-01
  8.25733052e-02 2.93257377e-01]
 [8.84396826e-01 2.40625693e-01 5.96098168e-01 3.94240074e-01
  3.40465807e-01 1.03846900e-01 3.96066561e-02 6.59874432e-02
  1.94963906e-01 6.46735604e-01]
 [5.51766198e-01 9.19683421e-01 5.55527471e-01 8.10970298e-01
  8.40675825e-01 4.82247653e-01 7.74210467e-01 1.20849386e-01
  1.92319174e-02 5.25271850e-01]
 [5.05846615e-01 6.44373077e-01 1.70196405e-01 2.00762231e-01
  8.47889889e-01 7.48813633e-01 2.64021718e-01 7.07920745e-01
  6.04821951e-01 8.77018057e-01]
 [3.43893966e-01 2.03590126e-01 7.40871853e-01 6.22687280e-01
  7.11067244e-02 6.40422594e-01 6.78749082e-01 1.27391196e-01
  7.25821162e-01 3.49446626e-01]
 [6.53701419e-02 8.06142093e-01 4.22474529e-01 9.90928993e-01
  9.38740586e-01 7.12664781e-01 5.08810064e-01 2.91143446e-01
  5.78035132e-01 7.69976517e-01]
 [7.50118013e-01 8.20094940e-01 1.55663245e-02 9.32628035e-01
  2.77692917e-01 8.54691780e-01 5.67096855e-01 6.13466530e-01
  5.34968939e-02 1.84959979e-01]
 [9.35986860e-01 1.10362837e-01 9.38576334e-01 9.50478594e-02
  6.08767382e-01 6.71124291e-01 7.36298962e-02 7.89499719e-01
  6.39721552e-01 7.16673657e-01]
 [3.51980029e-01 3.23525200e-01 3.67491942e-01 6.08015635e-01
  7.73275421e-01 8.17604181e-01 4.43681560e-01 6.12532656e-01
  4.26973586e-01 8.06412948e-01]
 [9.22146139e-01 4.46526530e-01 4.91615465e-01 9.92686861e-01
  7.41698235e-01 6.14902294e-01 6.36764484e-02 9.40175500e-01
  6.62883504e-01 5.86580197e-01]
 [3.78594933e-01 2.90727629e-01 8.67006864e-04 7.16513176e-01
  8.26190397e-01 5.86858980e-01 8.25245260e-01 6.50171620e-01
  9.12396877e-01 5.37803768e-01]
 [2.41892522e-01 7.07205965e-02 5.50143002e-01 5.76722795e-01
```

```python
# Training (very basic CBOW with numpy)
import numpy as np

# Initialize weights
W1 = np.random.rand(vocab_size, embed_dim)
W2 = np.random.rand(embed_dim, vocab_size)
learning_rate = 0.01
epochs = 1000

for epoch in range(epochs):
    loss = 0
    for context, target in data:
        # Convert words to indices
        context_ids = [word_to_ix[w] for w in context]
        target_id = word_to_ix[target]

        # Forward pass
        x = np.mean(W1[context_ids], axis=0)         # average embeddings
        z = np.dot(x, W2)
        y_pred = np.exp(z) / np.sum(np.exp(z))       # softmax

        # Calculate loss
        loss -= np.log(y_pred[target_id])
```

```
            # Backpropagation (gradient updates)
            y_pred[target_id] -= 1  # This is (y_hat - y)

            # Gradient for W2
            dW2 = np.outer(x, y_pred)
            W2 -= learning_rate * dW2

            # Gradient for W1 (only for context words)
            # d_Loss/d_x = W2 @ (y_hat - y)
            grad_hidden = np.dot(W2, y_pred)
            # Each context word embedding contributes to x, so share the gradie
            W1[context_ids] -= learning_rate * grad_hidden / len(context_ids)

        if epoch % 100 == 0:
            print(f'Epoch {epoch}, Loss: {loss:.4f}')
```

```
Epoch 0, Loss: 205.5006
Epoch 100, Loss: 167.7289
Epoch 200, Loss: 112.6938
Epoch 300, Loss: 59.7542
Epoch 400, Loss: 29.4665
Epoch 500, Loss: 15.6709
Epoch 600, Loss: 9.5136
Epoch 700, Loss: 6.4395
Epoch 800, Loss: 4.7098
Epoch 900, Loss: 3.6394
```

```
print("Similar words to 'process':")
similarities = np.dot(W1, W1[word_to_ix['process']])
sorted_words = np.argsort(-similarities)
for idx in sorted_words[:5]:
    print(ix_to_word[idx])
```

```
Similar words to 'process':
process
computational
by
pattern
manipulate
```

```
Start coding or generate with AI.
```