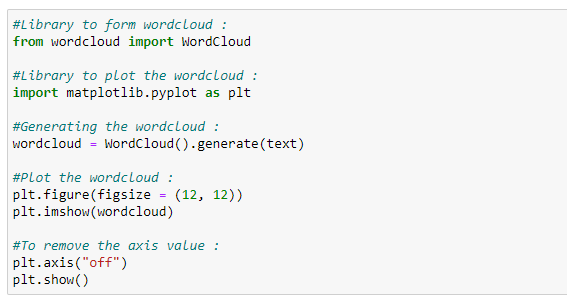
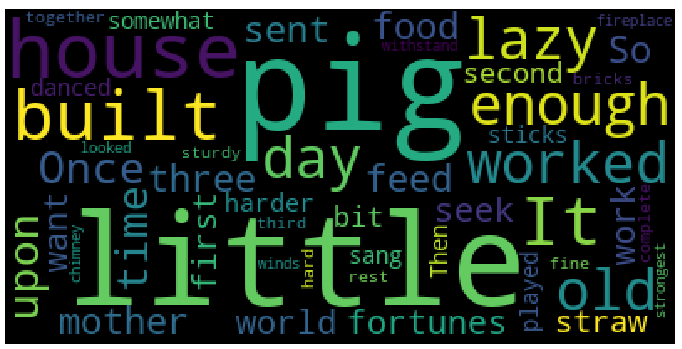
**Week 2: Word Cloud**

Word Cloud is a data visualization technique. In which words from a given text display on the main chart. In this technique, more frequent or essential words display in a larger and bolder font, while less frequent or essential words display in smaller or thinner fonts. It is a beneficial technique in NLP that gives us a glance at what text should be analyzed.

Properties:

1. **font\_path**: It specifies the path for the fonts we want to use.
2. **width**: It specifies the width of the canvas.
3. **height**: It specifies the height of the canvas.
4. **min\_font\_size**: It specifies the smallest font size to use.
5. **max\_font\_size:** It specifies the largest font size to use.
6. **font\_step**: It specifies the step size for the font.
7. **max\_words**: It specifies the maximum number of words on the word cloud.
8. **stopwords**: Our program will eliminate these words.
9. **background\_color:** It specifies the background color for canvas.
10. **normalize\_plurals**: It removes the trailing “s” from words.

Word Cloud Python Implementation:

Figure 31: Python code implementation of the word cloud.Figure 32: Word cloud example.

As shown in the graph above, the most frequent words display in larger fonts. The word cloud can be displayed in any shape or image.

For instance: In this case, we are going to use the following circle image, but we can use any shape or any image.

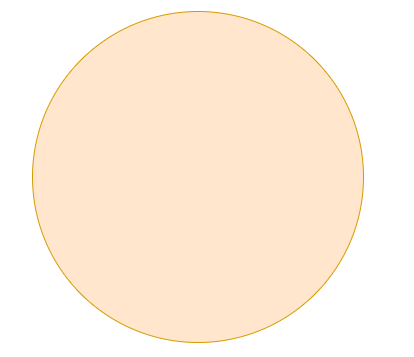


Figure 33: Circle image shape for our word cloud.

Word Cloud Python Implementation:

Figure 34: Python code implementation of the word cloud.

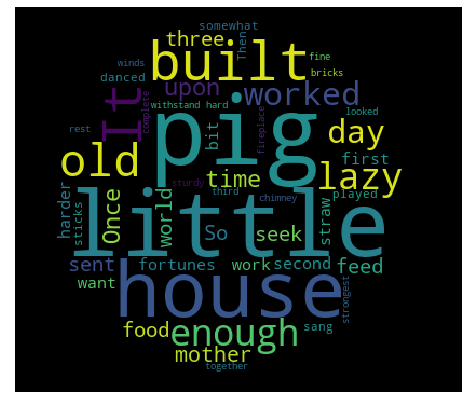


Figure 35: Word cloud with the circle shape.

As shown above, the word cloud is in the shape of a circle. As we mentioned before, we can use any shape or image to form a word cloud.

**Word CloudAdvantages:**

* They are fast.
* They are engaging.
* They are simple to understand.
* They are casual and visually appealing.

**Word Cloud Disadvantages:**

* They are non-perfect for non-clean data.
* They lack the context of words.

**Week 3: Stemming:**

We use Stemming to normalize words. In English and many other languages, a single word can take multiple forms depending upon context used. For instance, the verb “study” can take many forms like “studies,” “studying,” “studied,” and others, depending on its context. When we tokenize words, an interpreter considers these input words as different words even though their underlying meaning is the same. Moreover, as we know that NLP is about analyzing the meaning of content, to resolve this problem, we use stemming.

Stemming normalizes the word by truncating the word to its stem word. For example, the words “studies,” “studied,” “studying” will be reduced to **“studi,”** making all these word forms to refer to only one token. Notice that stemming may not give us a dictionary, grammatical word for a particular set of words.

Let’s take an example:

**a. Porter’s Stemmer Example 1:**

In the code snippet below, we show that all the words truncate to their stem words. However, notice that the stemmed word is not a dictionary word.



Figure 36: Code snippet showing a stemming example.

**b. Porter’s Stemmer Example 2:**

In the code snippet below, many of the words after stemming did not end up being a recognizable dictionary word.



Figure 37: Code snippet showing a stemming example.

**c. SnowballStemmer:**

SnowballStemmer generates the same output as porter stemmer, but it supports many more languages.



Figure 38: Code snippet showing an NLP stemming example.

**d. Languages supported by snowball stemmer:**

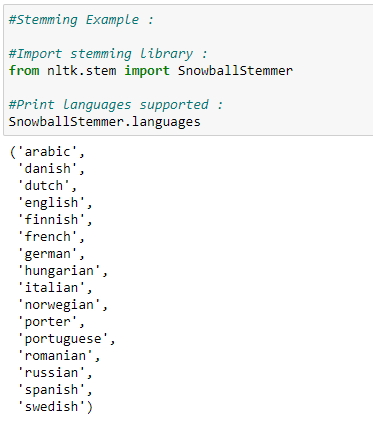
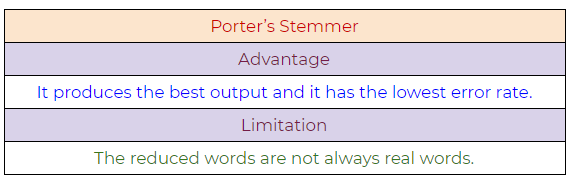


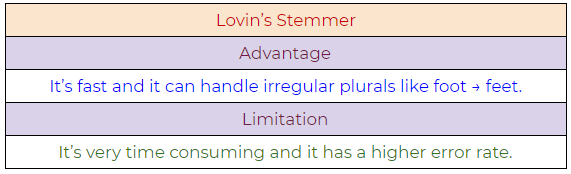
Figure 39: Code snippet showing an NLP stemming example.

Various Stemming Algorithms:

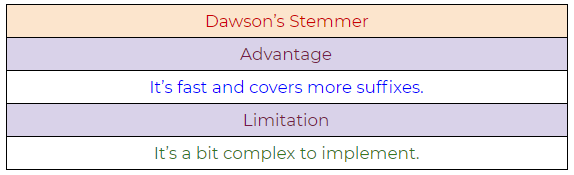
**a. Porter’s Stemmer:**

Figure 40: Porter’s Stemmer NLP algorithm, pros, and cons.

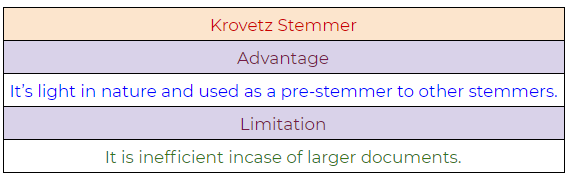
**b. Lovin’s Stemmer:**

Figure 41: Lovin’s Stemmer NLP algorithm, pros, and cons.

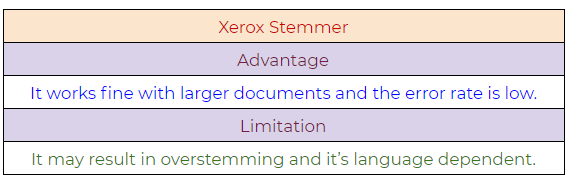
**c. Dawson’s Stemmer:**

Figure 42: Dawson’s Stemmer NLP algorithm, pros, and cons.

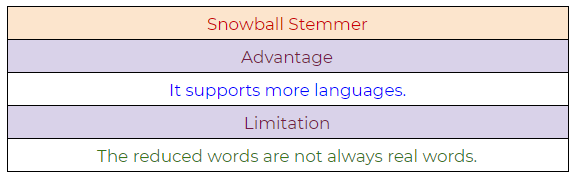
**d. Krovetz Stemmer:**

Figure 43: Krovetz Stemmer NLP algorithm, pros, and cons.

**e. Xerox Stemmer:**

Figure 44: Xerox Stemmer NLP algorithm, pros, and cons.

**f. Snowball Stemmer:**

Figure 45: Snowball Stemmer NLP algorithm, pros, and cons.