12: Text

Some examples related to the visualization of text data:

- A Visual Survey of Text Visualization Techniques (https://textvis.lnu.se/)
- On the Origin of Species (https://benfry.com/traces/)

For text data you will use `spaCy` and `wordcloud` to play with text data. `spaCy` is designed for analyzing text data, It is capable and fast.

```
pip install wordcloud spacy
```

To use spaCy, you also need to download models.

```
python -m spacy download en
```

SpaCy basics

In [8]:
```
import spacy
import wordcloud

nlp = spacy.load('en_core_web_sm')
```

Usually the first step of text analysis is *tokenization*, which is the process of breaking a document into "tokens". You can roughly think of it as extracting each word.

In [9]:
```
doc = nlp(u'Apple is looking at buying U.K. startup for $1 billion')

for token in doc:
    print(token)
```

```
Apple
is
looking
at
buying
U.K.
startup
for
$
1
billion
```

As you can see, it's not exactly same as `doc.split()` . You'd want to have `$` as a separate token because it has a particular meaning (USD). Actually, as shown in an example (https://spacy.io/usage/spacy-101#annotations-pos-deps (https://spacy.io/usage/spacy-101#annotations-pos-deps)), `spaCy` figures out a lot of things about these tokens. For instance,

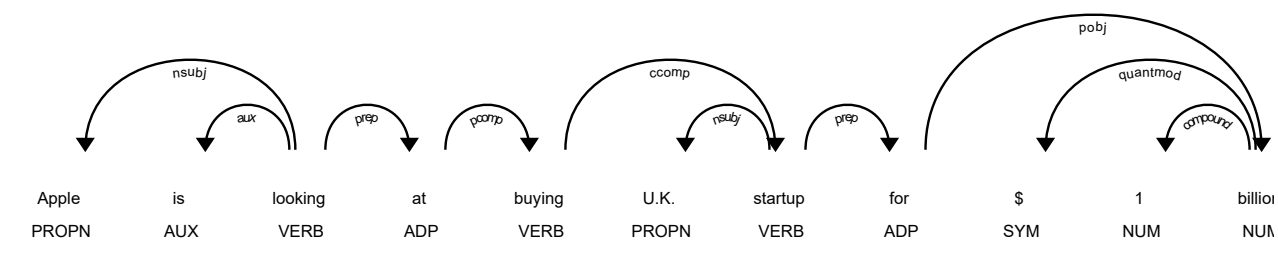
In [10]:
```
for token in doc:
    print(token.text, token.lemma_, token.pos_, token.tag_)
```

```
Apple Apple PROPN NNP
is be AUX VBZ
looking look VERB VBG
at at ADP IN
buying buy VERB VBG
U.K. U.K. PROPN NNP
startup startup VERB VBD
for for ADP IN
$ $ SYM $
1 1 NUM CD
billion billion NUM CD
```

It figured it out that `Apple` is a proper noun ("PROPN" and "NNP"; see here (https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html) for the part of speech tags).

`spaCy` has a visualizer too.

In [11]:
```
from spacy import displacy
displacy.render(doc, style='dep', jupyter=True, options={'distance': 100})
```



It even recognizes entities and can visualize them.

In [12]:
```
text = """But Google is starting from behind. The company made a late push
into hardware, and Apple’s Siri, available on iPhones, and Amazon’s Alexa
software, which runs on its Echo and Dot devices, have clear leads in
consumer adoption."""

doc2 = nlp(text)
displacy.render(doc2, style='ent', jupyter=True)
```

But Google ORG is starting from behind. The company made a late push

into hardware, and Apple’s Siri ORG , available on iPhones ORG , and Amazon ORG ’s Alexa

software, which runs on its Echo LOC and Dot devices, have clear leads in

consumer adoption.

Read a book

- You can use any books that you can find as a text file.

In [13]:
```
import requests

metamorphosis_book = requests.get('http://www.gutenberg.org/cache/epub/5200/pg5200.txt').content
```

In [14]:
```
metamorphosis_book[:1000]
```

Out[14]:
```
b'\xef\xbb\xbfThe Project Gutenberg eBook of Metamorphosis\r\n    \r\nThis ebook is for the use of anyone anywhere in the United States a
nd\r\nmost other parts of the world at no cost and with almost no restrictions\r\nwhatsoever. You may copy it, give it away or re-use it
under the terms\r\nof the Project Gutenberg License included with this ebook or online\r\nat www.gutenberg.org. If you are not located in
the United States,\r\nyou will have to check the laws of the country where you are located\r\nbefore using this eBook.\r\n\r\n*** This is
a COPYRIGHTED Project Gutenberg eBook. Details Below. ***\r\n***     Please follow the copyright guidelines in this file.      ***\r\n\r
\n\r\nTitle: Metamorphosis\r\n\r\nAuthor: Franz Kafka\r\n\r\nTranslator: David Wyllie\r\n\r\nRelease date: August 17, 2005 [eBook #5200]
\r\n                Most recently updated: April 28, 2021\r\n\r\nLanguage: English\r\n\r\n\r\n\r\n*** START OF THE PROJECT GUTENBERG EBOO
K METAMORPHOSIS ***\r\n\r\n\r\n\r\n\r\n\r\nMetamorphosis\r\n\r\nby Franz Kafka\r\n\r\nTranslated by David Wyllie\r\n\r\n\r\n\r\n\r\n\r\nI\r\n\r\n
\r'
```

- We have successfully loaded the book. You will probably want to remove the parts at the beginning and at the end that are not parts of the book if you are doing a serious analysis, but let's ignore them for now. Let's try to feed this directly into `spaCy` .

In [ ]:
```
# Try to run this cell. Is it giving exception? Why doesn't it work? Think about it for a moment before moving on!
# def nlp(text:str|Doc) but we pass bytes|any b'...'
try:
    doc_metamor = nlp(metamorphosis_book)
except TypeError as e:
    print("metamorphosis_book is not converted to string:", str(e))
```

```
---------------------------------------------------------------------------
ExtraData                                 Traceback (most recent call last)
Cell In[16], line 4
      1 # Try to run this cell. Is it giving exception? Why doesn't it work? Think about it for a moment before moving on!
      2 # we pass b'...' maybe expects string '...'?
      3 try:
----> 4     doc_metamor = nlp(metamorphosis_book)
      5 except TypeError as e:
      6     print("metamorphosis_book is not converted to string:", str(e))

File d:\Downloads\c10\.venv\lib\site-packages\spacy\language.py:1040, in Language.__call__(self, text, disable, component_cfg)
   1019 def __call__(
   1020     self,
   1021     text: Union[str, Doc],
   (...)
   1024     component_cfg: Optional[Dict[str, Dict[str, Any]]] = None,
   1025 ) -> Doc:
   1026     """Apply the pipeline to some text. The text can span multiple sentences,
   1027     and can contain arbitrary whitespace. Alignment into the original string
   1028     is preserved.
   (...)
   1038     DOCS: https://spacy.io/api/language#call
   1039     """
-> 1040     doc = self._ensure_doc(text)
   1041     if component_cfg is None:
   1042         component_cfg = {}

File d:\Downloads\c10\.venv\lib\site-packages\spacy\language.py:1133, in Language._ensure_doc(self, doc_like)
   1131     return self.make_doc(doc_like)
   1132 if isinstance(doc_like, bytes):
-> 1133     return Doc(self.vocab).from_bytes(doc_like)
   1134 raise ValueError(Errors.E1041.format(type=type(doc_like)))

File d:\Downloads\c10\.venv\lib\site-packages\spacy\tokens\doc.pyx:1362, in spacy.tokens.doc.Doc.from_bytes()

File d:\Downloads\c10\.venv\lib\site-packages\srsly\_msgpack_api.py:27, in msgpack_loads(data, use_list)
     25 # msgpack-python docs suggest disabling gc before unpacking large messages
     26 gc.disable()
---> 27 msg = msgpack.loads(data, raw=False, use_list=use_list)
     28 gc.enable()
     29 return msg

File d:\Downloads\c10\.venv\lib\site-packages\srsly\msgpack\__init__.py:85, in unpackb(packed, **kwargs)
     83         object_hook = functools.partial(decoder, chain=object_hook)
     84     kwargs["object_hook"] = object_hook
---> 85 return _unpackb(packed, **kwargs)

File d:\Downloads\c10\.venv\lib\site-packages\srsly\msgpack\_unpacker.pyx:213, in srsly.msgpack._unpacker.unpackb()

ExtraData: unpack(b) received extra data.
```

On encodings

Why fo you get this error? What does it mean? It says `nlp` function expects `str` type but we passed `bytes` .

In [17]:
```
type(metamorphosis_book)
```

Out[17]:
```
bytes
```

Indeed, the type of `metamorphosis_book` is `bytes` . But as we have seen above, we can see the book contents? What's going on?

The problem is that a byte sequence is not yet a proper string until we know how to decode it. A string is an abstract object and we need to specify an encoding to write the string into a file. For the same carachter you can have different encodings. This is a really important (and confusing) topic, as this is beyond the scope of the course, please read this nice post about encoding: http://kunststube.net/encoding/ (http://kunststube.net/encoding/)

Project Gutenberg uses `utf-8` encoding. So let's decode the byte sequence into a string.

In [22]:
```
metamorphosis_book_str = metamorphosis_book.decode('utf-8')
metamorphosis_book_str[:100]
```

Out[22]:
```
'\ufeffThe Project Gutenberg eBook of Metamorphosis\r\n    \r\nThis ebook is for the use of anyone anywhere in'
```

In [23]:
```
type(metamorphosis_book_str)
```

Out[23]:
```
str
```

Shall we try again?

In [24]:
```
doc_metamor = nlp(metamorphosis_book_str)
```

In [25]:
```
words = [token.text for token in doc_metamor
         if token.is_stop != True and token.is_punct != True]
```

Let's count!

In [26]:
```
from collections import Counter

Counter(words).most_common(5)
```

Out[26]:
```
[('\r\n', 1883),
 ('Gregor', 298),
 ('\r\n\r\n', 143),
 ('room', 131),
 ('sister', 101)]
```

a lot of newline characters and multiple spaces. A quick and dirty way to remove them is split & join. The idea is that you split the document using `split()` and then join with a single space ` `. Can you implement it and print the 10 most common words?

In [29]:
```
doc_metamor = nlp(' '.join(metamorphosis_book_str.split()))
words = [token.text for token in doc_metamor
         if token.is_stop != True and token.is_punct != True]
Counter(words).most_common(10)
```

Out[29]:
```
[('Gregor', 298),
 ('room', 131),
 ('sister', 101),
 ('father', 99),
 ('door', 87),
 ('Project', 86),
 ('Gutenberg', 86),
 ('mother', 84),
 ('work', 74),
 ('way', 64)]
```

Let's keep the object with word count.

In [30]:
```
word_cnt = Counter(words)
```

Wordclouds

In [31]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

Can you check out the `wordcloud` package documentation and create a word cloud from the word count object that we created from the book above and plot it?

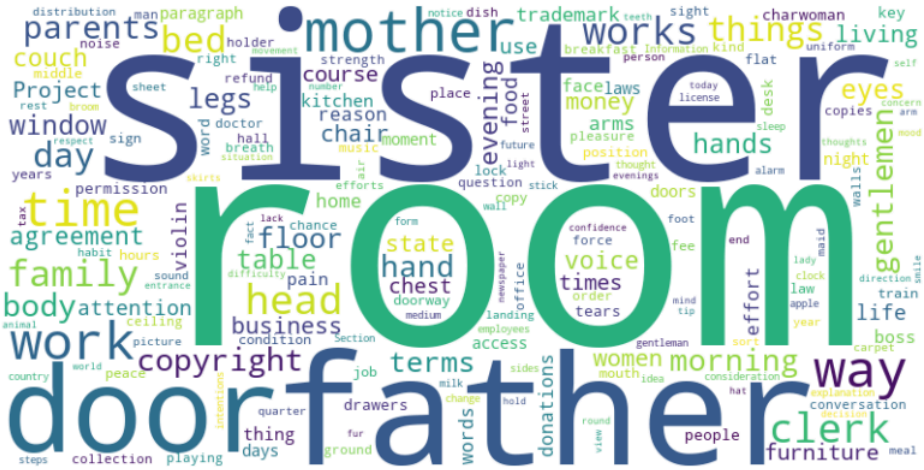
In [32]:
```
wcloud = wordcloud.WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(word_cnt)
wcloud
```

Out[32]:
```
<wordcloud.wordcloud.WordCloud at 0x225df0a2b90>
```

In [35]:
```
plt.figure(figsize=(10, 5))
plt.imshow(wcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Q: Can you create a word cloud for a certain part of speech, such as nouns, verbs, proper nouns, etc. (pick one)?

In [41]:
```
words = [token.text for token in doc_metamor
         if token.is_stop != True and token.is_punct != True and token.pos_=='NOUN']
word_cnt = Counter(words)
wcloud = wordcloud.WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(word_cnt)
plt.figure(figsize=(10, 5))
plt.imshow(wcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



In [ ]:
```
```