Johannes Gutenberg-Universität Mainz

Institut für Geschichts- und Kulturwissenschaften

Seminar: Natural Language Processing

Lecturer: Demival Vasques Filho

Winter Semester 2021/2022

Asking Questions In
Alice's Adventures In Wonderland

Concerning Nouns And Named Entities

07.02.2022:

Jan Eberhardt

Schwertstraße 11

67549 Worms

J-F-E@web.de

Digitale Methodik in den Geistes- und Kulturwissenschaften, 1. Semester

Table of Contents

Table of Figures

Abstract

This paper, as the whole project, centers around "Alice's Adventures in Wonderland" by Lewis Carroll. The intention was to find out what questions are asked in the book, but more specifically what nouns and named entities do the questions contain.

Anaconda and Jupyter Notebook were used as working environment. To extract the questions from the text, different regular expressions were tried out. The open-source library spaCy was used to extract the nouns from the questions, as well as to search for named entities. The latter was also done with BookNLP to get a comparison of effectivity.

Although having a few problems at the beginning with extracting just the questions from the text, in the end the used nouns and named entities were found, listed, and visualized as word clouds. Both used programs to detect them weren't right to 100%, but BookNLP had a higher effectivity than spaCy.

1 Introduction

In a first thought about a text I could use for the project, the works of Hildegard von Bingen came to mind, but since I didn't get an answer from one of the heads of the Hildegard project of the Digital Academy, I jumped to my backup thought for the project: "Alice's Adventures in Wonderland", to whom I have a strong relationship because of the White Rabbit.

As source material, "Alice's Adventures in Wonderland" was used from a repository by a user named Phillip Johnsen, who got a positive comment by another user, who studied regular expression and used this version of the text for their work, comparing it to the version published by the Gutenberg Project, which had a few errors.[1]

To clear this text from unnecessary white spaces and line breaks, it was first run through Python executions, before searching for every "?" used in the text and trying to extract the appertaining questions with the re module.

The questions were eventually extracted into a different file that was imported into Jupyter Notebook to work with spaCy and BookNLP. To search for the nouns, the method of part of speech tagging, which is a spaCy feature, was used. A differ-

[1] Drjoms, on Johnsen, P. 2021. alice_in_wonderland.txt. San Francisco (CA): GitHub. [accessed 2022 02 05]. URL: https://gist.github.com/phillipj/4944029?permalink_comment_id=3186160#gistcomment-3186160

ent feature, the automated NER (named entity recognition) , was used to get to know which named entities there are in Wonderland. To compare the effectivity of this spaCy tool, a different one called BookNLP by David Bamman that also provides the extraction of named entities of a text, was used.

For the visualization of the results, the Word Cloud repository by Andreas Mueller provided pretty word clouds in form of Alice herself.

Screenshots of the work in progress were made and are used as references in this paper.

2 Methodology

Tuomo Hiippala provided a very good course for basic text manipulation with Python, as well as working with texts using spaCy. Several parts of the used code were provided by this course.

Hiippala also wrote a small part about using regular expressions, that were used in this project to look for every "?" in "Alice's Adventures in Wonderland":

"Python allows using regular expressions through its re module. [...] Capturing these patterns would require defining more complex regular expressions, which are harder to write. Their complexity is, however, what makes regular expressions so powerful, but at the same time, learning how to use them takes time and patience. [...] In practice, coming up with regular expressions that cover as many matches as possible is particularly hard." [2]

Concerning the method of how to get the nouns out of a text, Afham Fardeen and again Hiippala wrote about the spaCy part of speech tags:
"Spacy provides a bunch of POS tags such as NOUN (noun), PUNCT (punctuation), ADJ(adjective), ADV(adverb), etc. It has a trained pipeline and statistical models which enable spaCy to make classification of which tag or label a token belongs to. For example, a word following 'the' in English is most likely a noun.

Spacy also provides a fine-grained tag that further categorizes a token in different sub-categories. For example, when a word is an adjective it further categorizes

[2] Hiippala, Tuomo (2021) Applied Language Technology: NLP for the Humanities. In David Jurgens, Varada Kolhatkar, Lucy Li, Margot Mieskes and Ted Pedersen (eds) Proceedings of the Fifth Workshop on Teaching NLP. Association for Computational Linguistics, 46–48. DOI: 10.18653/v1/2021.teachingnlp-1.5

it as JJR (comparative adjective), JJS (superlative adjective), or AFX (affix adjective)."[3]

"*spaCy* provides two types of part-of-speech tags, *coarse* and *fine-grained*, which are stored under the attributes pos_ and tag_, respectively.

We can access the attributes of a Python object by inserting the *attribute* after the *object* and separating them with a full stop, e.g. token.pos_.

To access the results of POS tagging, let's loop over the *Doc* object doc and print each *Token* and its coarse and fine-grained part-of-speech tags.

In contrast to coarse part-of-speech tags, the fine-grained tags also encode grammatical information. The tags for verbs, for example, are distinguished by aspect and tense. "[4]

What is the used method of Named Entity Recognition? Mareike Schumacher, who is a known personality in the Digital Humanities community, wrote in her article concerning NER:

"*Named Entity Recognition (NER)* ist ein Verfahren, mit dem klar benennbare Elemente (z.B. Namen von Personen oder Orten) in einem Text automatisch markiert werden können. *Named Entity Recognition* wurde im Rahmen der computerlinguistischen Methode des *Natural Language Processing (NLP)* entwickelt, bei der es darum geht, natürlichsprachliche Gesetzmäßigkeiten maschinenlesbar aufzubereiten. "[5]

Tuomo Hiippala adds, especially referring to the NER tool used by spaCy: "spaCy can recognise the named entities annotated in the OntoNotes 5 corpus, such as persons, geographic locations and products, to name but a few examples."[6]

"This doc property is used for the named entities in the document. If the entity recognizer has been applied, this property will return a tuple of named entity span objects.", writes the website tutorialspoint concerning the doc.ents property in spaCy.[7]

[3] Fardeen, Afham (2021). **Tutorial on Spacy Part of Speech (POS) Tagging, machinelearningknowledge.ai**. Available at: https://machinelearningknowledge.ai/tutorial-on-spacy-part-of-speech-pos-tagging/ (Accessed: 05 February 2022).

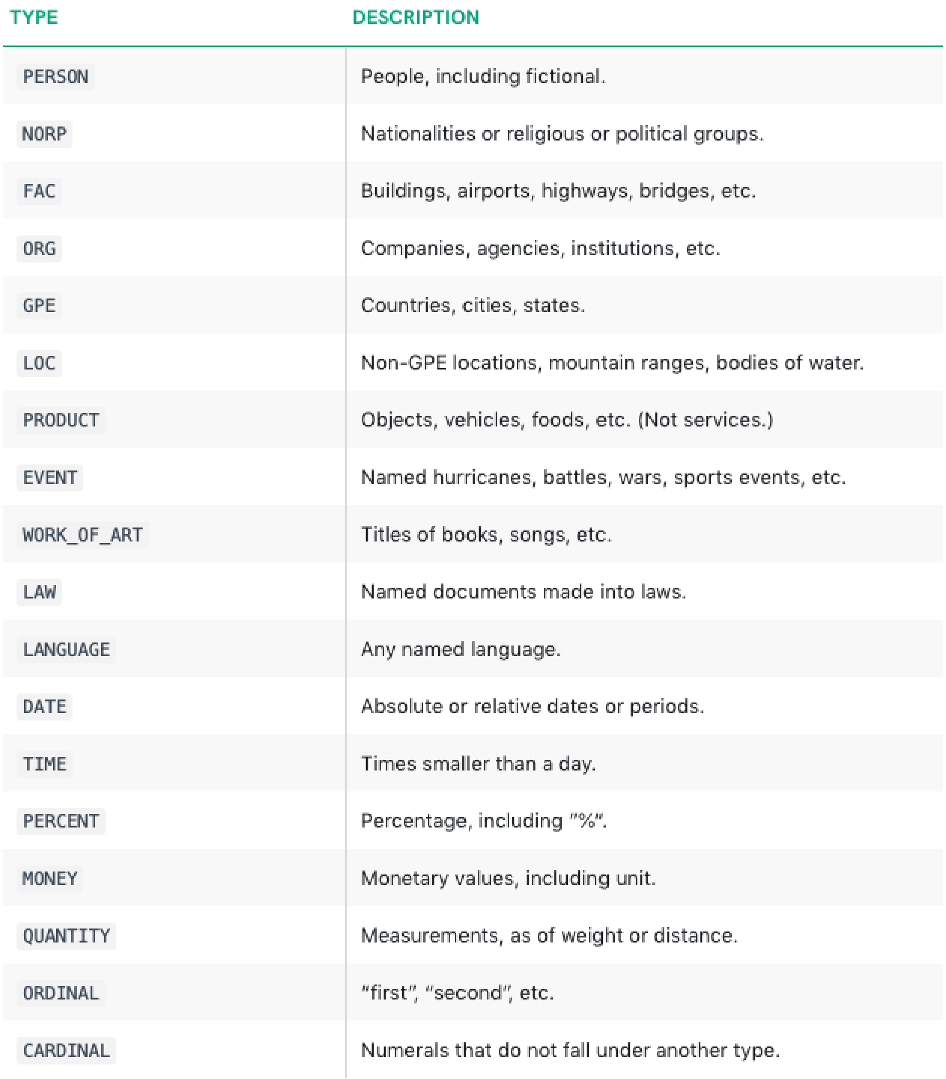
[4] Hiippala, Tuomo (2021) Applied Language Technology: NLP for the Humanities. In David Jurgens, Varada Kolhatkar, Lucy Li, Margot Mieskes and Ted Pedersen (eds) Proceedings of the Fifth Workshop on Teaching NLP. Association for Computational Linguistics, 46–48. DOI: 10.18653/v1/2021.teachingnlp-1.5

[5] Schumacher, M. (2018) *Named Entity recognition (NER)*, *forTEXT*. Available at: https://fortext.net/routinen/methoden/named-entity-recognition-ner (Accessed: 05 February 2022).

[6] Hiippala, Tuomo (2021) Applied Language Technology: NLP for the Humanities. In David Jurgens, Varada Kolhatkar, Lucy Li, Margot Mieskes and Ted Pedersen (eds) Proceedings of the Fifth Workshop on Teaching NLP. Association for Computational Linguistics, 46–48. DOI: 10.18653/v1/2021.teachingnlp-1.5

[7] spaCy - Doc.ents Property. Available at: https://www.tutorialspoint.com/spacy/spacy_doc_ents.htm (Accessed: 05 February 2022).

Here is a list of all possible entities to recognize with spaCy:



TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.
PERCENT	Percentage, including ”%“.
MONEY	Monetary values, including unit.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	“first”, “second”, etc.
CARDINAL	Numerals that do not fall under another type.

Figure 1: Named entities in the OntoNotes 5 corpus[8]

[8] Sanidhya (2020) Named Entity Recognition with spaCy, Medium. Available at: https://medium.com/analytics-vidhya/named-entity-recognition-with-spacy-2ecfa4114162 (Accessed: 05 February 2022).

To get a comparison of effectivity of the spaCy model, we chose the BookNLP pipeline as second model to analyze the questions. David Bamman writes on his repository:

“The entity annotation layer covers six of the ACE 2005 categories in text:

- People (PER): *Tom Sawyer, her daughter*
- Facilities (FAC): *the house, the kitchen*
- Geo-political entities (GPE): *London, the village*
- Locations (LOC): *the forest, the river*
- Vehicles (VEH): *the ship, the car*
- Organizations (ORG): *the army, the Church*

The entity tagging model within BookNLP is trained on an annotated dataset of 968K tokens, including the public domain materials in LitBank and a new dataset of ~500 contemporary books, including bestsellers, Pulitzer Prize winners, works by Black authors, global Anglophone books, and genre fiction.”[9]

Finally, for visualization, the Word Cloud repository by Andreas Mueller was used. Coincidentally, besides the standard word cloud in form of a rectangle, he also provided a mask in form of Alice and the White Rabbit, which was of course used immediately.[10]

[9] Bamman, D. 2021. Booknlp. San Francisco (CA): GitHub. [accessed 2022 02 05]. URL: https://github.com/booknlp/booknlp#entity-annotations

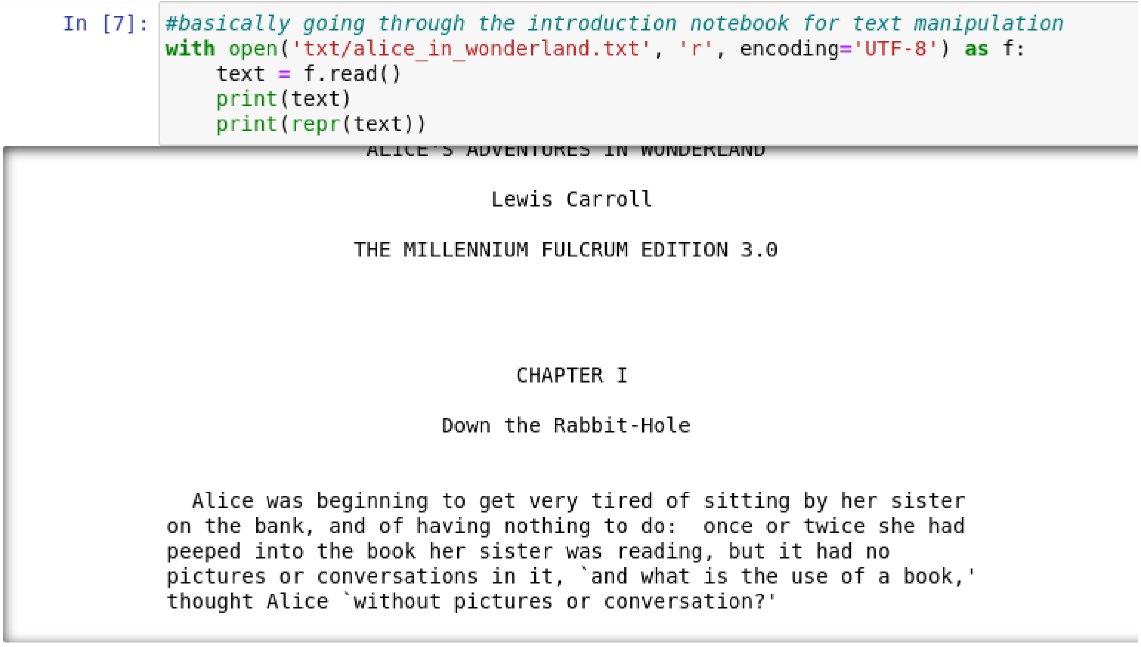
[10] Mueller, A. 2021. word_cloud. San Francisco (CA): GitHub. [accessed 2022 02 05]. URL: https://github.com/amueller/word_cloud

3 Data and Results

As source material for “Alice’s Adventures in Wonderland”, Phillip Johnsen's repository was used, because of a comment on it, which made optimistic:
“drjoms commented on 23 Feb 2020:
thanks. was studying regular expression. decided to use the book as sample material. Gutenberg version seemed to add some characters at end of line, which rendered me in state of frustration. your version restored my sanity. thanks!”[11]

Reading the imported text file, it seems to be well-structured, just as in the book itself (see Fig. 2). To work with the text, we have to get rid of unnecessary white spaces and line breaks, but doing so creates some unnatural backspaces, for example before the Apostrophe in a Genitive (see Fig. 3). Trying to replace backspaces with white spaces doesn’t help, though.

One way to look for question marks in the text is by defining a pattern in the regular expression module re in Python and using it on the text. This is one of the closest ways compared to the very easy “ctrl + f” version on your keyboard. There are 202 question marks in the text (see Fig. 4).

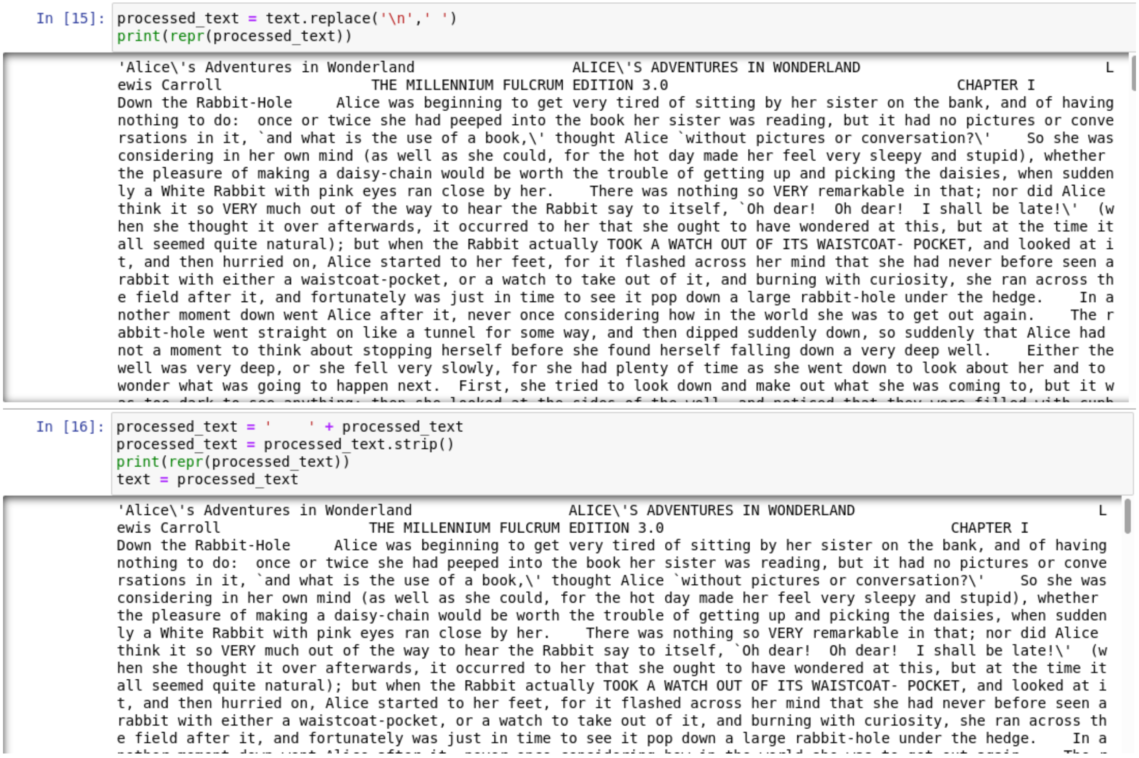


```
In [7]: #basically going through the introduction notebook for text manipulation
        with open('txt/alice_in_wonderland.txt', 'r', encoding='UTF-8') as f:
            text = f.read()
            print(text)
            print(repr(text))
```

```
                        ALICE'S ADVENTURES IN WONDERLAND

                                  Lewis Carroll

                        THE MILLENNIUM FULCRUM EDITION 3.0




                                    CHAPTER I

                                Down the Rabbit-Hole


  Alice was beginning to get very tired of sitting by her sister
on the bank, and of having nothing to do:  once or twice she had
peeped into the book her sister was reading, but it had no
pictures or conversations in it, `and what is the use of a book,'
thought Alice `without pictures or conversation?'
```

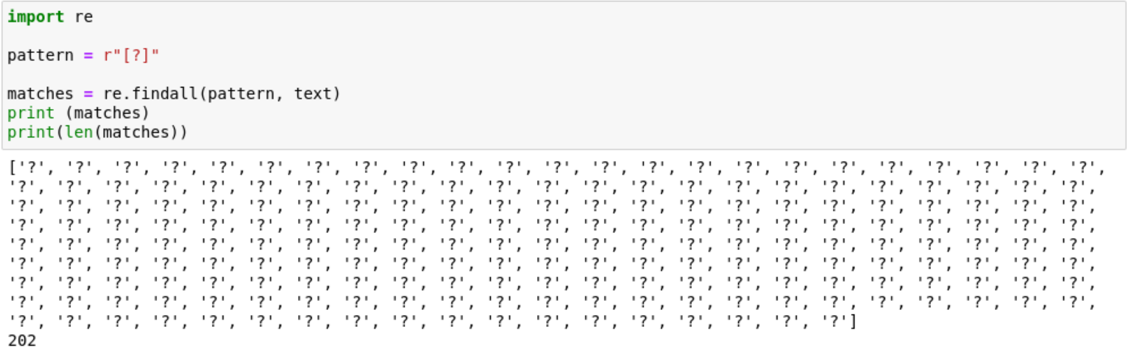
Figure 2: Text file of the source

[11] Drjoms, on Johnsen, P. 2021. alice_in_wonderland.txt. San Francisco (CA): GitHub. [accessed 2022 02 05]. URL: https://gist.github.com/phillipj/4944029?permalink_comment_id=3186160#gistcomment-3186160



```
In [15]: processed_text = text.replace('\n',' ')
         print(repr(processed_text))

'Alice\'s Adventures in Wonderland                  ALICE\'S ADVENTURES IN WONDERLAND                    L
ewis Carroll                THE MILLENNIUM FULCRUM EDITION 3.0                           CHAPTER I
Down the Rabbit-Hole     Alice was beginning to get very tired of sitting by her sister on the bank, and of having
nothing to do:  once or twice she had peeped into the book her sister was reading, but it had no pictures or conve
rsations in it, `and what is the use of a book,\' thought Alice `without pictures or conversation?\'    So she was
considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether
the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when sudden
ly a White Rabbit with pink eyes ran close by her.    There was nothing so VERY remarkable in that; nor did Alice
think it so VERY much out of the way to hear the Rabbit say to itself, `Oh dear!  Oh dear!  I shall be late!\'  (w
hen she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it
all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT- POCKET, and looked at i
t, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a
rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across th
e field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.    In a
nother moment down went Alice after it, never once considering how in the world she was to get out again.    The r
abbit-hole went straight on like a tunnel for some way, and then dipped suddenly down, so suddenly that Alice had
not a moment to think about stopping herself before she found herself falling down a very deep well.    Either the
well was very deep, or she fell very slowly, for she had plenty of time as she went down to look about her and to
wonder what was going to happen next.  First, she tried to look down and make out what she was coming to, but it w

In [16]: processed_text = '     ' + processed_text
         processed_text = processed_text.strip()
         print(repr(processed_text))
         text = processed_text

'Alice\'s Adventures in Wonderland                  ALICE\'S ADVENTURES IN WONDERLAND                    L
ewis Carroll                THE MILLENNIUM FULCRUM EDITION 3.0                           CHAPTER I
Down the Rabbit-Hole     Alice was beginning to get very tired of sitting by her sister on the bank, and of having
nothing to do:  once or twice she had peeped into the book her sister was reading, but it had no pictures or conve
rsations in it, `and what is the use of a book,\' thought Alice `without pictures or conversation?\'    So she was
considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether
the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when sudden
ly a White Rabbit with pink eyes ran close by her.    There was nothing so VERY remarkable in that; nor did Alice
think it so VERY much out of the way to hear the Rabbit say to itself, `Oh dear!  Oh dear!  I shall be late!\'  (w
hen she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it
all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT- POCKET, and looked at i
t, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a
rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across th
e field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.    In a
```

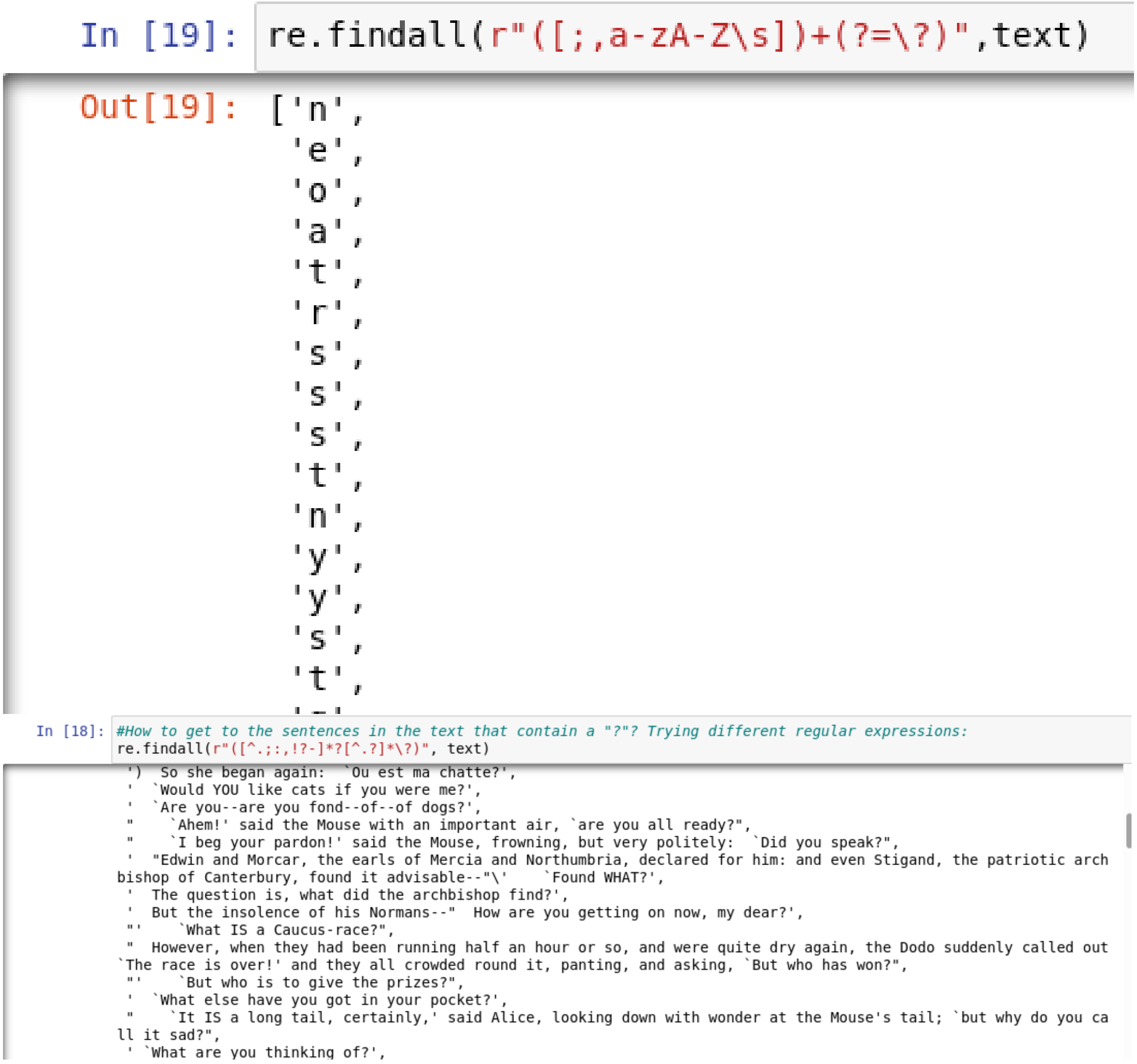
Figure 3: processed text



```
import re

pattern = r"[?]"

matches = re.findall(pattern, text)
print (matches)
print(len(matches))

['?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?',
'?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?',
'?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?',
'?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?',
'?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?',
'?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?',
'?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?',
'?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?',
'?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?', '?']
202
```

Figure 4: looking for “?” in the text

In further use of regular expressions, different patterns were used to look for every sentence ending with a question mark and extracting them. But the expressions returned either no sentences at all or too many sentences (see Fig. 5). In the worst cases, Jupyter Notebook was freezing while executing the command. The best result that was possible can be seen on lower part of figure 5. But that was not good enough.



```
In [19]: re.findall(r"([;,a-zA-Z\s])+(?=\?)",text)

Out[19]: ['n',
          'e',
          'o',
          'a',
          't',
          'r',
          's',
          's',
          's',
          't',
          'n',
          'y',
          'y',
          's',
          't',
```

```
In [18]: #How to get to the sentences in the text that contain a "?"? Trying different regular expressions:
         re.findall(r"([^.;:,!?-]*?[^.?]*\?)", text)
```

```
 ')  So she began again:  `Ou est ma chatte?',
  '  `Would YOU like cats if you were me?',
  '  `Are you--are you fond--of--of dogs?',
  "    `Ahem!' said the Mouse with an important air, `are you all ready?",
  "    `I beg your pardon!' said the Mouse, frowning, but very politely:  `Did you speak?",
  '  "Edwin and Morcar, the earls of Mercia and Northumbria, declared for him: and even Stigand, the patriotic arch
bishop of Canterbury, found it advisable--"\'    `Found WHAT?',
  '  The question is, what did the archbishop find?',
  '  But the insolence of his Normans--"  How are you getting on now, my dear?',
  "'    `What IS a Caucus-race?",
  "  However, when they had been running half an hour or so, and were quite dry again, the Dodo suddenly called out
`The race is over!' and they all crowded round it, panting, and asking, `But who has won?",
  "'    `But who is to give the prizes?",
  '  `What else have you got in your pocket?',
  "    `It IS a long tail, certainly,' said Alice, looking down with wonder at the Mouse's tail; `but why do you ca
ll it sad?",
  ' `What are you thinking of?',
```

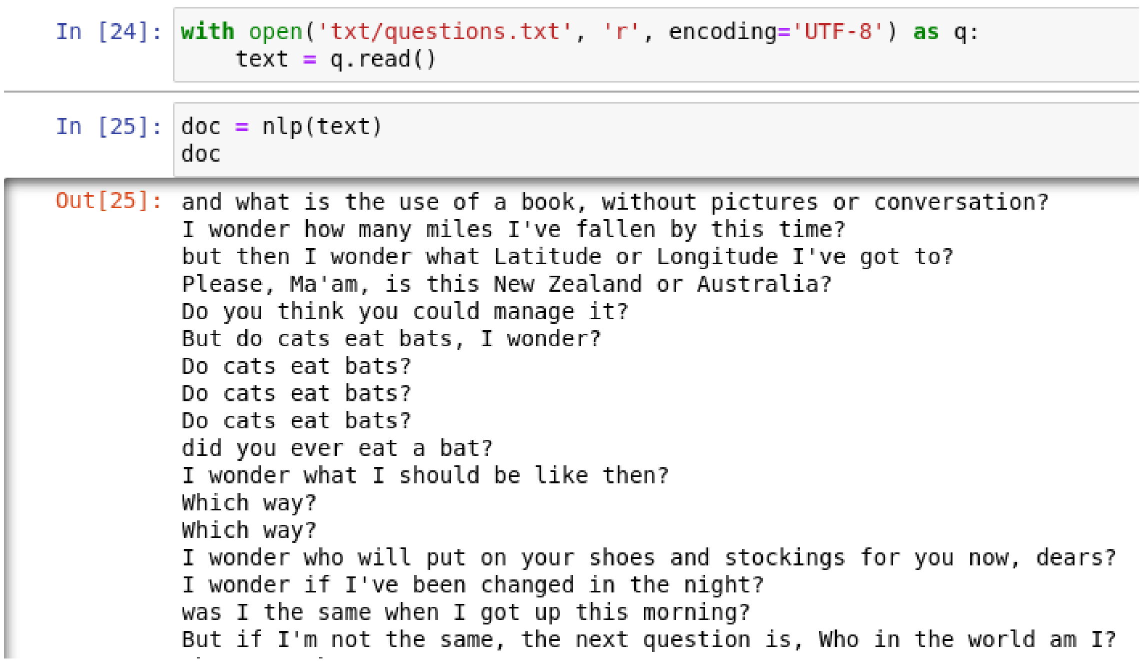
Figure 5: trying to get every sentence ending with a “?”

To get a better result for the actual question of what nouns and named entities there are in the questions, a by-hand-extraction using the old “ctrl + f” method was necessary. So, a new file with all the questions perfectly lined up was imported into Jupyter Notebook and the doc object of spaCy, annotated to the nlp object, to be able to process the natural language in the questions (see Fig. 6).

The first thing we do with this doc is that we loop over each token (item in the list) and print the *coarse* and *fine-grained* part of speech tags. As we can see, there are different tags for nouns (see Fig. 7), and there is also a different category of nouns: the proper nouns (with the fine-grained tag NNP).

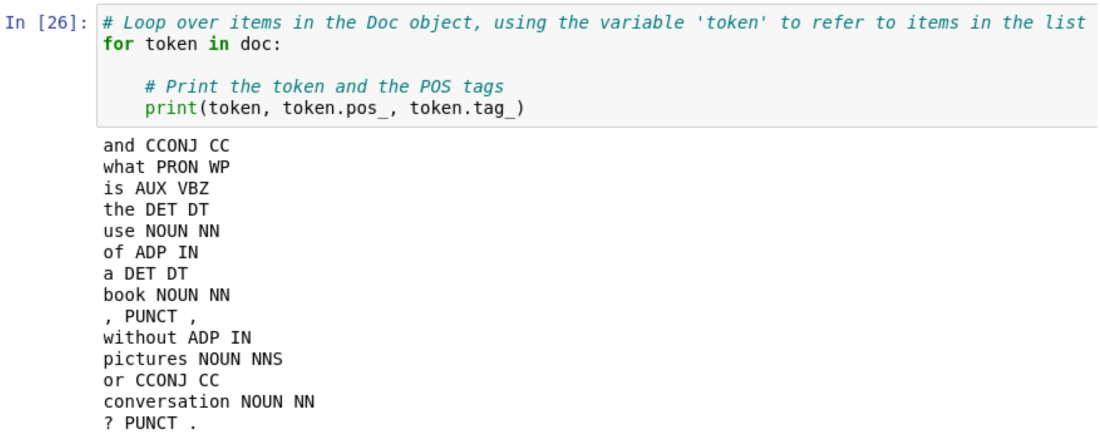
To be able to get a word cloud visualization at the end (see Fig. 9), we have to make sure that every noun, no matter if it is a noun in singular form, a noun in plural form or a proper noun, write it into a new file (see Fig. 8).

There were a few problems with spaCy running through the questions and extracting the nouns, though: Sometimes the word was actually an adjective, such as “queer”, for example. It also detected the French words “Ou” (Where) and “ma” (my) as nouns, as well as word “grins”, which is a verb.



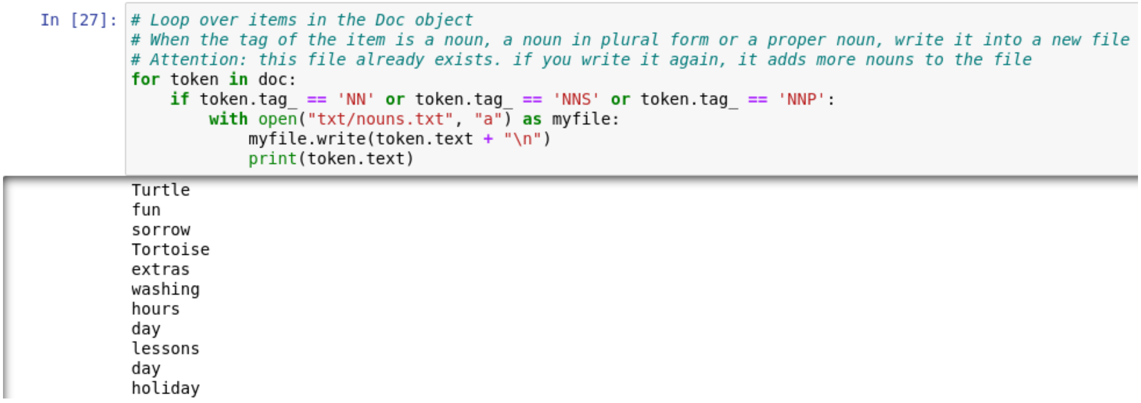
```
In [24]: with open('txt/questions.txt', 'r', encoding='UTF-8') as q:
             text = q.read()

In [25]: doc = nlp(text)
         doc

Out[25]: and what is the use of a book, without pictures or conversation?
         I wonder how many miles I've fallen by this time?
         but then I wonder what Latitude or Longitude I've got to?
         Please, Ma'am, is this New Zealand or Australia?
         Do you think you could manage it?
         But do cats eat bats, I wonder?
         Do cats eat bats?
         Do cats eat bats?
         Do cats eat bats?
         did you ever eat a bat?
         I wonder what I should be like then?
         Which way?
         Which way?
         I wonder who will put on your shoes and stockings for you now, dears?
         I wonder if I've been changed in the night?
         was I the same when I got up this morning?
         But if I'm not the same, the next question is, Who in the world am I?
```

Figure 6: Import the file with all the questions lined up



```
In [26]: # Loop over items in the Doc object, using the variable 'token' to refer to items in the list
         for token in doc:

             # Print the token and the POS tags
             print(token, token.pos_, token.tag_)

         and CCONJ CC
         what PRON WP
         is AUX VBZ
         the DET DT
         use NOUN NN
         of ADP IN
         a DET DT
         book NOUN NN
         , PUNCT ,
         without ADP IN
         pictures NOUN NNS
         or CCONJ CC
         conversation NOUN NN
         ? PUNCT .
```

Figure 7: get the POS tags for every token in the doc



```
In [27]: # Loop over items in the Doc object
         # When the tag of the item is a noun, a noun in plural form or a proper noun, write it into a new file
         # Attention: this file already exists. if you write it again, it adds more nouns to the file
         for token in doc:
             if token.tag_ == 'NN' or token.tag_ == 'NNS' or token.tag_ == 'NNP':
                 with open("txt/nouns.txt", "a") as myfile:
                     myfile.write(token.text + "\n")
                     print(token.text)
```

```
Turtle
fun
sorrow
Tortoise
extras
washing
hours
day
lessons
day
holiday
```

Figure 8: get every noun in the doc

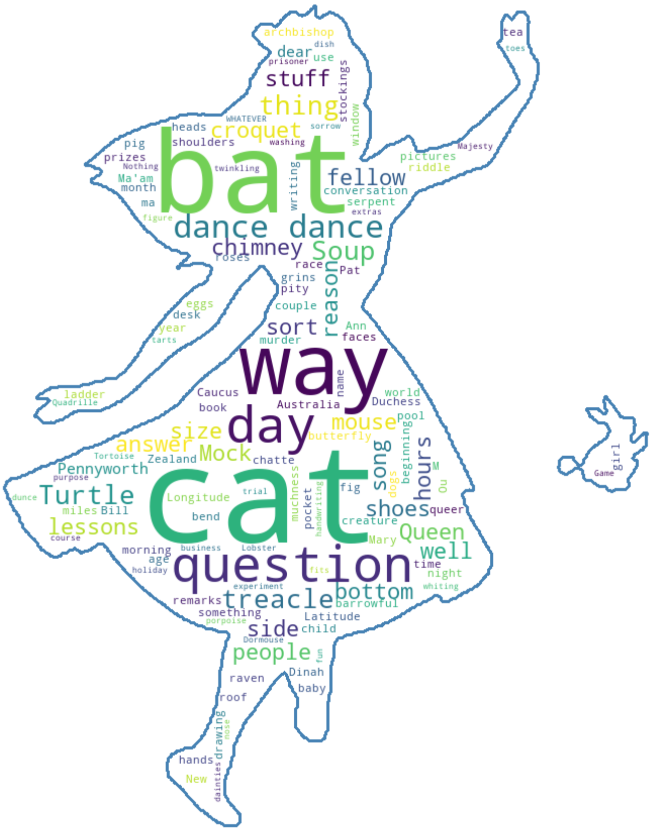


Figure 9: Word Cloud – Nouns

Concerning the named entities there are similar problems, but first things first:

To get the entities form the spaCy model, we take each ent object and print it together with its label. For a better visualization, we can also use a tool from spaCy called displacy (see Fig. 10).

What we can see on Figure 10 is that there are only 22 named entities in the questions, more likely the spaCy model just found 22, and some of them are definitely not labeled rightly:

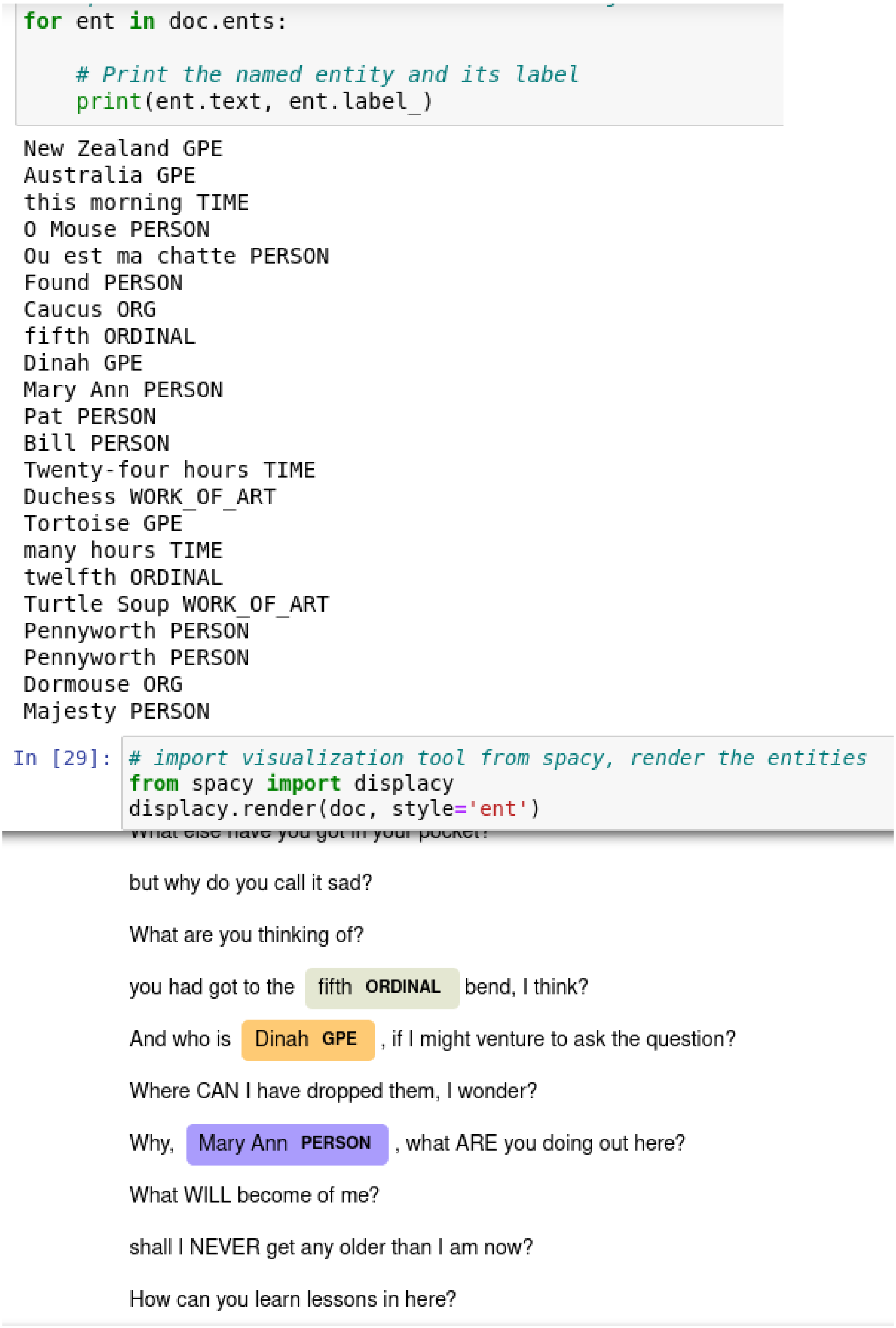
While Australia and New Zealand are rightly tagged as geopolitical entities, Dinah is Alice's cat, and the tortoise is also a person. "O Mouse" was declared a name, recognizing "O" as first name and "Mouse" as last name, also the whole French question "Ou est ma chatte?" was declared a person, which makes absolutely no sense. Finally the duchess was declared a work of art, while she is a person and the Dormouse, one of the most famous figures of the book, was declared an organization.

To get the comparison to spaCy, let's look at Figure 11. Here, the nominals and proper nouns were extracted from the originally created entity file by BookNLP. There are 41 named entities, many more than spaCy got.

While Dinah, the Duchess and the Dormouse all get a correct label with person, "O Mouse" stays the same. The French is ignored, but something else is going wrong: "another figure of the Lobster Quadrille" is not a person, since the Lobster Quadrille is a dance, the figure is a dance move. "This New Zealand" also sounds strange, and the fact that the article is always included, if there is one, is one that can be discussed for its rightness.

So, the results clearly show that the included NER tool from spaCy isn't as effective as the BookNLP tool, that – as you can already see in its name – is specialized in analyzing books, while spaCy isn't specialized in analyzing anything specific.

Finally, for the visualization effect, we also get Alice shaped word clouds for both results concerning the named entity recognition (see Fig. 12 & 13).



```
for ent in doc.ents:

    # Print the named entity and its label
    print(ent.text, ent.label_)
```

```
New Zealand GPE
Australia GPE
this morning TIME
O Mouse PERSON
Ou est ma chatte PERSON
Found PERSON
Caucus ORG
fifth ORDINAL
Dinah GPE
Mary Ann PERSON
Pat PERSON
Bill PERSON
Twenty-four hours TIME
Duchess WORK_OF_ART
Tortoise GPE
many hours TIME
twelfth ORDINAL
Turtle Soup WORK_OF_ART
Pennyworth PERSON
Pennyworth PERSON
Dormouse ORG
Majesty PERSON
```

```
In [29]: # import visualization tool from spacy, render the entities
         from spacy import displacy
         displacy.render(doc, style='ent')
```

What else have you got in your pocket?

but why do you call it sad?

What are you thinking of?

you had got to the fifth ORDINAL bend, I think?

And who is Dinah GPE , if I might venture to ask the question?

Where CAN I have dropped them, I wonder?

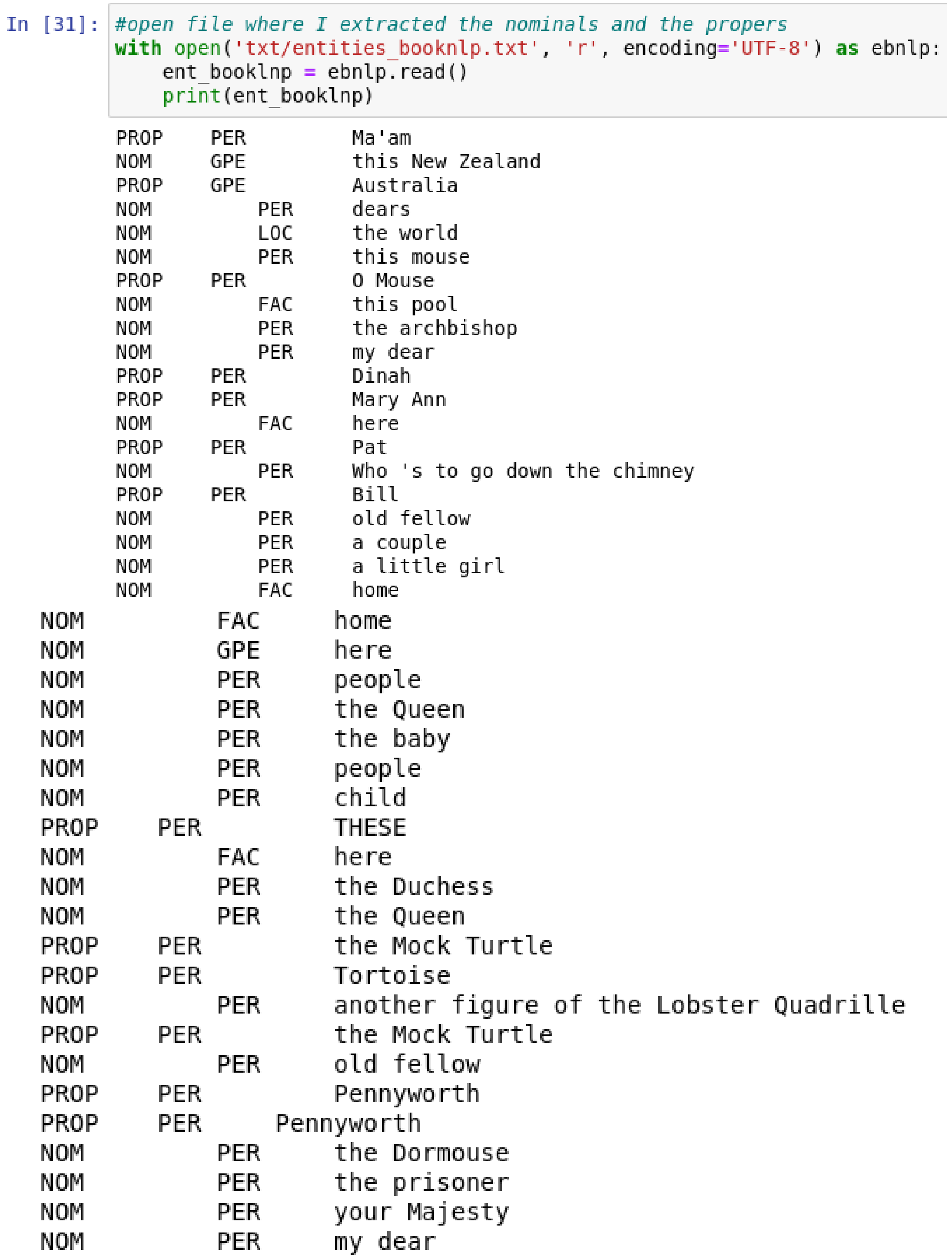
Why, Mary Ann PERSON , what ARE you doing out here?

What WILL become of me?

shall I NEVER get any older than I am now?

How can you learn lessons in here?

Figure 10: get every named entity in the doc (including displacy)



```
In [31]: #open file where I extracted the nominals and the propers
         with open('txt/entities_booknlp.txt', 'r', encoding='UTF-8') as ebnlp:
             ent_booklnp = ebnlp.read()
             print(ent_booklnp)
```

```
PROP    PER         Ma'am
NOM     GPE         this New Zealand
PROP    GPE         Australia
NOM         PER     dears
NOM         LOC     the world
NOM         PER     this mouse
PROP    PER         O Mouse
NOM         FAC     this pool
NOM         PER     the archbishop
NOM         PER     my dear
PROP    PER         Dinah
PROP    PER         Mary Ann
NOM         FAC     here
PROP    PER         Pat
NOM         PER     Who 's to go down the chimney
PROP    PER         Bill
NOM         PER     old fellow
NOM         PER     a couple
NOM         PER     a little girl
NOM         FAC     home
```

```
NOM         FAC     home
NOM         GPE     here
NOM         PER     people
NOM         PER     the Queen
NOM         PER     the baby
NOM         PER     people
NOM         PER     child
PROP    PER         THESE
NOM         FAC     here
NOM         PER     the Duchess
NOM         PER     the Queen
PROP    PER         the Mock Turtle
PROP    PER         Tortoise
NOM         PER     another figure of the Lobster Quadrille
PROP    PER         the Mock Turtle
NOM         PER     old fellow
PROP    PER         Pennyworth
PROP    PER     Pennyworth
NOM         PER     the Dormouse
NOM         PER     the prisoner
NOM         PER     your Majesty
NOM         PER     my dear
```

Figure 11: every named entity with BookNLP

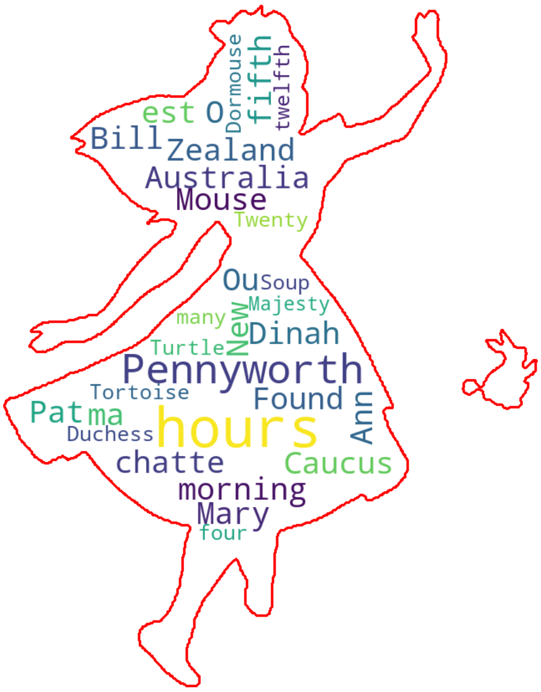


Figure 12: Word Cloud – Entities from SpaCy

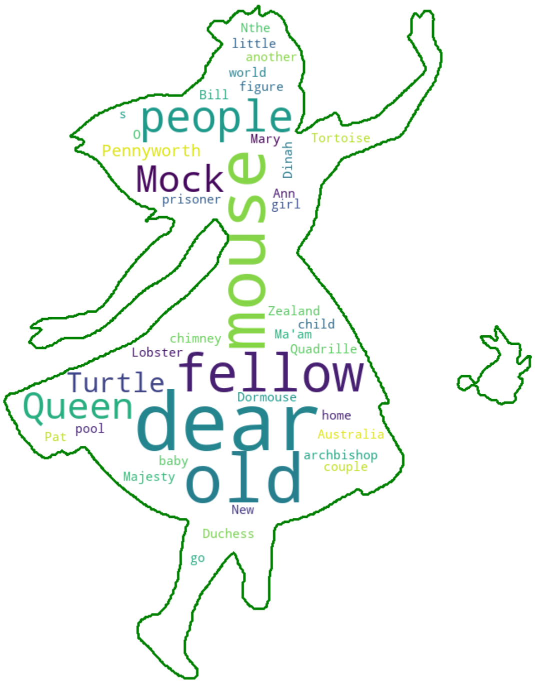


Figure 13: Word Cloud – Entities from BookNLP

4 Discussion

As Hiippala stated, using regular expressions as search tool to extract certain patterns out of a text can get quite difficult, and that’s exactly what happened here. To get the expertise in this topic, more time would be needed. For a text like “Alice’s Adventures in Wonderland”, it wasn’t that problematic to extract the questions by hand, but for larger corpora it would be definitely helpful to get a good method running smoothly. A way to start would be to understand what questions are on a professional linguistic level.

There were a few problems with spaCy going through the questions and extracting the nouns, as well as the named entities. It was a disappointment to see that spaCy didn’t recognize the Dormouse as an organization and not as a person, for it is clearly one of the main characters at least at the famous tea party.

To better detect the French words, and not recognize them as a person, we could maybe include the spaCy multilanguage model, or even the small French one, even though it would take spaCy longer to initialize at the beginning.

Why is BookNLP better in recognizing named entities than spaCy? This might be the case, because BookNLP always uses the large data model as default. [12] For spaCy that also might help: using the large language model for English.

A larger language model would be useful for both models actually, since none of them had an accuracy of 100% in detecting named entities. The corpora (OntoNotes 5 corpus for spaCy and the big self-trained corpus for BookNLP) aren’t trained well enough to properly detect nouns or named entities in a fantasy text from the 19th century.

For now, you have to get over each noun or named entity by yourself to be sure the program got it right or wrong.

[12] Bamman, D. 2021. Booknlp. San Francisco (CA): GitHub. [accessed 2022 02 05]. URL: https://github.com/booknlp/booknlp#entity-annotations

5 Conclusion and a Look into the Future

As we could see in the results and as already stated in the discussion part: Both programs to detect named entities (spaCy and bookNLP) didn’t work very well, even though BookNLP worked much better, but that might be the case, because BookNLP uses the larger language model by default. SpaCy also had problems detecting nouns. The corpora of both programs don’t seem to be trained enough. There is much potential, but the models aren’t trained well enough to get certain words right. As said before, you basically have to get over each noun or named entity by yourself to be sure the program got it right or wrong. Or you train a new model, specifically adjusted to your text. But the programs alone don’t have the power to manage the language by themselves yet.

Concerning this topic, it might be interesting to see the difference between the data we used now and the data that would be produced if we used a different language model, larger or also including multiple languages.

Interesting question for further studies could be: How many questions (and respectively what nouns and named entities) are there in each chapter of “Alice’s Adventures in Wonderland”? Also finding out who is asking who about what would be very interesting. The visualization could be done with the graph modeling tool neo4j.

List of References

Bamman, D. 2021. Booknlp. San Francisco (CA): GitHub. [accessed 2022 02 05]. URL: https://github.com/booknlp/booknlp#entity-annotations

Drjoms, on Johnsen, P. 2021. alice_in_wonderland.txt. San Francisco (CA): GitHub. [accessed 2022 02 05]. URL: https://gist.github.com/phillipj/4944029?permalink_comment_id=3186160#gistcomment-3186160

Fardeen, Afham (2021). Tutorial on Spacy Part of Speech (POS) Tagging, machinelearningknowledge.ai. Available at: https://machinelearningknowledge.ai/tutorial-on-spacy-part-of-speech-pos-tagging/ (Accessed: 05 February 2022).

Hiippala, Tuomo (2021) Applied Language Technology: NLP for the Humanities. In David Jurgens, Varada Kolhatkar, Lucy Li, Margot Mieskes and Ted Pedersen (eds) Proceedings of the Fifth Workshop on Teaching NLP. Association for Computational Linguistics, 46–48. DOI: 10.18653/v1/2021.teachingnlp-1.5

Mueller, A. 2021. word_cloud. San Francisco (CA): GitHub. [accessed 2022 02 05]. URL: https://github.com/amueller/word_cloud

Sanidhya (2020) Named Entity Recognition with spaCy, Medium. Available at: https://medium.com/analytics-vidhya/named-entity-recognition-with-spacy-2ecfa4114162 (Accessed: 05 February 2022).

Schumacher, M. (2018) Named Entity recognition (NER), forTEXT. Available at: https://fortext.net/routinen/methoden/named-entity-recognition-ner (Accessed: 05 February 2022).

spaCy - Doc.ents Property. Available at: https://www.tutorialspoint.com/spacy/spacy_doc_ents.htm (Ac-cessed: 05 February 2022).