**Argument**

Digital Humanties is by now an well-established academic field. For some outsiders, it might trigger the impression that it is simply a hip buzzword, or merely a passing trend. The trend definetly isn’t going to fade out any time soon, since the digital age has remained unrelenting since it’s dawn in the late 80’s / early 90’s. As for the other part of our binomial, the Humanities, to attempt to compress its rich history and tradition in a paper’s argument would be mean either naivety or arrogance. However, the humanties cannot be possible divorced by technological progress. The classic, widely known example: Gutenberbg’s press - it revolutionized the capacity of text production and circulation. Therefore, we arrive at the conclusion that it fundamentally changed how humanist scholars organized their activity, since the text was always their fundamental work object.

It is also true that the humanties evolved to be more specialized and branched out, to suit the needs of the increasing body of knowledge that it produced, but also the demands for accuracy and veracity (how ancient and medieval chroniclers depicted events vs. the modus operandis of contemporary historians). This shift is itself indicative of a change of paradigm, from the generalist, Rennaisance scholar to the modern, highly specialized in his niche modern researcher. In this context, Digital Humanities is itself a field of subfields, an umbrella term, similar to how the Humanities, in their current understanding hold the same significance. Therefore, the adjective “digital” is a consequence of the Humanities continuing to exist in the current paradigm.

In Humanistic research, as I’ve already stated, text is usually the main object of study. There are also connected domains that mainly study real-life, palpable objects (archeology, numismatics), or social realities and human behavior (anthropology, psychology neurosciences). But before these modern disciplines and their methods, the humanist tradition lies on fundamental texts: *The Bible*, Homer’s *Illiad* and *Odyssey*, Plato’s *Republic*, Aristotle’s *Poetics* and so on.

Nowadays, information is being generated and circulated at overwhelming rates. The internet is an ever-shifting environment where different media coexist and combine. Patrik Svenson’s provides insights on this phenomenon:

One important and apparent consequence of increased digitalization and, in particular, the web, is highly increased access to and availability of different types of content and media. Some of this content is born-analogue and much of it is born-digital. Increasingly, but not necessarily, these expressions are media rich, polytextual and mixed. [Schnapp & Shanks 2009, 147] discuss "fungibility" — the gathering of many types of content (moving image, text, music, 3D-design, database, graphical detail, virtual walk-through etc.) into a single environment — as the core of digital mediation. Content can accordingly be infinitely manipulated and remobilized without loss.

[Svenson, Patrik. The Landscape of Digital Humanities in the Digital Humanties Quaterly, Providence Vol.4. 2010]

As shown by Svenson, content produced in the digital age is heterogeneous by nature. In contrast, established literary works, fiction or non-fiction, are just blocks of texts, ocassionally with images or illustrations. Them existing in digital format doesn’t mean a fundamental transformation at the content level. However, this format opens up the usage of digital tools and, therefore, new angles of viewing established works. Svenson (2010) provides his take on what the general aim of digital humanties should be:

The digital humanities comprise the study of what happens at the intersection of computing tools with cultural artefacts of all kinds. This study begins where basic familiarity with standard software ends. It probes how these common tools may be used to make new knowledge from our cultural inheritance and from the contemporary world. [Ibidem]

What I focus on in this paper are textual cultural aretfacts: literary texts that I enjoy and am familiar with, with the aim of revisting them from a different angle using a machine. Although it was theorized by various scholars and given multiple defintions, distant reading, as a general, collective term, encompasses the usage of various digital tools applied to literary corpora.

In this paper, I am employing several distant reading tools with aim of tracing and comparing the works of established authors in the Sci-fi genre. The selected corpora are run trough BookNLP, a a python-based data-mining pipeline, specialized for narrative fiction. Other Python libraries are then employed for combing and visualizing the obtained data, as well as a web-based platform, Voyant.

**1. Introduction**

* 1. **The authors**

I selected for analysis the works of three wildely influental authors of sci-fi (altough the work of the first two were attached to different labels also, they remain under the sci-fi umbrella in broad sense): H.P. Lovecraft, Philip K. Dick, William Gibson. Their work spawned significant followings and fandoms, which arguably became subgenres of their own under the sci-fi umbrella. The authors’ work extends well beyond their fiction into philosophical essays, social critique, and correspondance with other authors and thinkers.

**Howard Phillips Lovecraft** (1890-1937) devoleped a shared universe between his various short stories and novellas, governed by an impressive mythos of his own creation. This wicked imagery is a direct extension of his phiolosophy: cosmicism, which states that human life holds no significance against the vast, hostile and incomprehensible universe that hosts it. His work is to a great extent a reflection of his hypondriac personality and precarious health that was a constant throughout his life. Unfortunately, much of his social anxieties fell under the spectre of racism and white supremacism, views that were quite common in his time. He spawned a genre that is now known as cosmic horror, or even Lovecraftian horror, when his mythos is directly referenced.

**Philip Kindred Dick**’s (1952-1982) fiction is usually centered on his characters struggle to comply with their given reality. His fears reflect the unstable political enviorment of his youth during the cold war, which mainly involved distrust in the government and the looming nuclear menance. His heavy use of substances, which eventually brough him to an early end, fueled a dystopic, convoluted imagery, where a contrived reality is controlled by entities (govermental superstructures, aliens, artificial intelligence) beyond human understanding. Besides sci-fi, his work has been labelled as paranoid fiction or even philosophical fiction, but nonetheless, the technological absurdity is the engine of his fiction.

**William Ford Gibson** (1948) had a later literary debut, he is actually 4 years younger than Philip K. Dick. Both visionary and foreboding, his work, just as Lovecraft’s, devoleped over a shared universe that entails the definitive aeshtetics and socio-politics of cyberpunk. This subgenre has been succinctly defined by the “high-tech, low-life” tennet. Gibson warns about a world where human life becomes increasingly expandable, crumbling under the pervasive and domineering technology that surparssed its creators. His characters deal with loss of identity, as their consciousness is being fragmented by systems outside of their understanding.

**1.2. Selected corpora**

My aim was to collect a balanced corpus across the authors. For each, I selected one longer work – a short novel, and 4 short stories/novellas, resulting in a corpus of ~100.000 words per author.

For **H.P. Lovecraft**, I have selected the novel *At the mountains of madness* (1931), and 4 short stories/novellas: *Call of Cthulhu* (1928), *The Colour out of Space* (1927), *The shadow over Innsmouth* (1931), *The Whisperer in Darkness* (1930).

The Lovecraft corpus has 5 documents with 119,555 total words, 11,873 unique word forms and an average sentence length of 25.7 words.



1.1. Lovecraft Voyant overview

For **Philip K. Dick**, my selection includes the novel *Do androids dream of electric sheep?* (1968) and 4 short stories: *Electric Ant* (1969), *Faith of our Fathers (1967)*, *Impostor* (1953), *The minority report (1956)*.

This corpus has 5 documents with 100,430 total words, 9,202 unique word forms and an average of sentence length of 10.3 words.



1.2. Philip K. Dick Voyant overview

**William Gibson**’s corpus consists of the novel *Neuromancer* (1984) and the 4 short stories:

*Burning Chrome* (1982), *Hinterlands* (1981), *New Rose Hotel* (1986), *Johnny Mnemonic* (1981).

This corpus has 5 documents with 103,272 total words, 10,909 unique word forms and an average sentence length of 12.9 words. Intrestingly, the longest text, Neuromancer, has the shortest words/sentences ratio (10.7)



1.3. Gibson Voyant overview

Each text of each author will be run trough the BookNLP pipeline.

**1.3. BookNLP**

The digital tool that I am employing is a Python-based NLP pipeline suggestively named BookNLP. It was devoloped by David Bamman, associate professor in the School of Information at Berkeley, University of California. The GitHub documentation page lays out the pipeline’s capacities:

BookNLP is a natural language processing pipeline that scales to books and other long documents (in English), including:

* Part-of-speech tagging
* Dependency parsing
* Entity recognition
* Character name clustering (e.g., "Tom", "Tom Sawyer", "Mr. Sawyer", "Thomas Sawyer" -> TOM\_SAWYER) and coreference resolution
* Quotation speaker identification
* Supersense tagging (e.g., "animal", "artifact", "body", "cognition", etc.)
* Event tagging
* Referential gender inference (TOM\_SAWYER -> he/him/his)

[Bamman, David, BookNLP Github repository, 2022 <https://github.com/booknlp/booknlp>]

W.J.B. Mattingly, researcher at Smithosian Data Science Lab and United States Holocaust Memorial Museum, used the pipeline extensively in his work and also created learning resources for making it more accesible (Youtube tutorial series and a website that holds examples and lessons). He explains the rationale behind converting the pipeline, which was originally conceived in Java, to Python:

BookNLP is a new Python library created by David Bamman. It was originally created as a Java library in 2014 under the same name, BookNLP by David Bamman, Ted Underwood, and Noah Smith (see, David Bamman, Ted Underwood and Noah Smith, “A Bayesian Mixed Effects Model of Literary Character,” ACL 2014). While Java is a powerful coding language, both in speed and ease-of-use, not many digital humanists code in Java primarily. I suspect (I want to emphasize I could be wrong) the reason for the Python library was to address the larger Python-coding community both in general and specifically within the digital humanities.

[Mattingly, William. *Introduction to BookNLP*, 2022. [booknlp.pythonhumanities.com](https://booknlp.pythonhumanities.com/booknlp.pythonhumanities.com)]

Mattingly comments on the pipeline’s intended usage and capacities:

Both the documentation and this textbook emphasize the word *large* here. The reason? Because most language models do not perform well with larger documents. Old RNN-based language models had a hard time remembering earlier words and while newer transformer-based models, such as BERT, have a larger memory and can look forwards and backwards, the size of the input they can take in is only 512 words. For larger documents, therefore, different solutions (and libraries) should be considered. This is where BookNLP comes in. It also addresses several problems associated with books and larger documents, such as:

* Characters (and people) are referenced by different names. BookNLP solves this problem with name clustering and coreference resolution. This is a task in NLP where we try and find all uses a name and correctly assign them to the same identifier, such as Harry, Harry Potter, and Mr. Harry Potter all being the same person, Harry Potter.
* An adjacent problem is referential gender inferencing. Like coreference resolution, often times in a book or larger document, a person will be referred to as a pronoun. This is where referential gender inferencing comes in. This allows a user to correctly assign the antecedent or postcedent to the correct pronoun. When done successfully, this also allows you to make decisions about the gender of the character or person based on how they are referenced in the text. Because this task is so delicate, given the delicate nature of assigning gender, BookNLP fortunately gives users the data with each pronoun used to reference a character and also includes non-binary pronouns.
* Another issue is quotation speaker identification. This is when we need to understand who is speaking, so that we can correctly link characters to their dialogues. It is possible to do this with spaCy, but it is extremely difficult to do well. BookNLP does a remarkable job of handling this problem and it does it with a fair degree of accuracy, from what I have seen. [Ibidem]

**1.4 Related work**

BookNLP has been employed research across the literary domain, with results that showcase both its strenghts, as well as its limits. In this section I will be briefing a few that caught my attention.

Eve Kraicer and Andrew Piper, in their 2019 paper *Social Characters: The Hierarchy of Gender in Contemporary English-Language Fiction*, analyse the prevalance, visibility and and different types of social connectivity of female characters across a vast aray of english-language fiction published between 2001 and 2015. In this endeavour, they use BookNLP’s character recognition functionality:

How visible are women in novels? There are different ways in which one might try to answer this question. We conceptualize the visibility of women in novels as a problem of hierarchial ranking – who gets mentioned more in a novel and how is the gender of characters distribuited in terms of frequency of mentions? Rather than look at the overall ratios of gendered pronouns for example, we are interested in the ranked ordering of men and women as entities within a novel’s fictional universe. To measure this, we rank characters by the number of mentions a character recieves over the course of a novel as detected by BookNLP (including proper names, aliases, and pronouns associated with character) and then assign that character a gender based on BookNL’s prediction. We then calculate social visibilty as a ratio of genders across four related sets: all characters; the top twenty most important characters; the novel’s main character; and finally the top-pair of characters. In addition to comparing these distributions across genres, we also examine the extent to which women authors influence the visibility of women characters in novels. These measures are designed to mirror previous research on women’s underrepresentation in other social and cultural spaces.

[Kraicer, Piper, Social Characters: The Hierarchy of Gender in Contemporary English-Language Fiction, p. 9 and 10, 2019]

A group of researches used BookNLP to extract and analyse character networks from a text corpus of J.R.R Tolkien’s Legendarium. Their approach and result were published in the paper *One Graph to Rule Them All: Using NLP and Graph Neural Networks to analyse Tolkien’s Legendarium*. Their approach is different from the previous research that I briefed, in that they choose to use entity recognition only when characters are mentioned by their name and ignore pronoun mentions, as BookNLP isn’t accurate enough in this regard, for their purposes:

Entity recognition refers to the task of detecting all references to entities (e.g., characters, location) in a text corpus. These references can either be explicitly named references (e.g. “Bilbo Baggins, “Smaug”), noun phrases (e.g., “the hobbit”, “the dragon”) or pronouns (e.g. “she, “they”). BookNLP uses an entity annotation model that has been trained on a large annotated data set [27] to identifiy named entities, noun phrases as well as pronoun references. After these references have been detected, in a next step coreference resolution can be applied, which is a very hard task in general [28] and is especially hard in the context of literary texts due to the high variation of references used and the very long texts [29, 30]. Confirming this view, our initial analyses revealed that the performance of BookNLP’s coreference resolution, which was trained on a data set of annotated coreferences [31] was not satisfactory when applying it to our corpus. We thus decided to focus on named references, and resolve these using a set of simply manually-created disambiguation rules (e.g. “Sam” -> “Sam Gamgee”, “Peregrin” -> “Pippin”). Although this approach may yield a low recall (i.e. there are many unidentified coreferences since pronouns and noun phrases are not considered), we find that this coreference resoluion yields high precision (i.e. almost all resolved coreferences that we inspected manually were correct). We found this approach preferable over a “full” coreference resolution for two reasons: First, considering our focus on character co-occurences that would harm our analyses of graph learning techniques. Second, our corpus of Tolkien’s Legendarium is special in the sense that it has a large number of named references, which give rise to rich character networks despite limiting our view to named references.

[Perri et al., One Graph to Rule Them All: Using NLP and Graph Neural Networks to analyse Tolkien’s Legendarium]

In *HarryMotions – Classifying Relationships In Harry Potter based on Emotion Analysis*, a simmilar approach to Tolkien’s Legendarium is employed on the Harry Potter series.

**2. Results and disscusion**

**2.1 Obtaining the data**

First step is running the pipeline for each seperate text of the corpora. After installing the pipeline package in Python, I run it using the code from the documentation:

from booknlp.booknlp import BookNLP

model\_params={

        "pipeline":"entity,quote,supersense,event,coref",

        "model":"big"

    }

booknlp=BookNLP("en", model\_params)

author = 'dick'

work = 'minority\_report'

# Input file to process

input\_file='works' + '/' + author + '/' + work + '/' + work + '.txt'

# Output directory to store resulting files in

output\_directory='works' + '/' + author + '/' + work + '/'

# File within this directory will be named ${book\_id}.entities, ${book\_id}.tokens, etc.

book\_id=work

booknlp.process(input\_file, output\_directory, book\_id)

The model\_params variable is used to instruct the pipeline on the data should it look for and the file it should generate for storage of said data. In this form, I run the full pipeline, generating for each text the following files:



2.1. Output files

There are 4 TSV files, one JSON file and one HTML file (altough this is not apparent from the extensions, as this extensions are part of the pipeline’s convention).

4 TSV files:

1. The **.tokens** file: this file contains a list of each each token in the text, meaning each word and punctuation mark that appears in the text. Within this tabular data system, each token is registred with its location in the text file (paragraph ID, sentence ID, token ID within document, token ID within sentence), its form (actual form, lemma) and its morpho-syntactic proprieties ( POS tag, fine\_POS tag, dependency relation, syntactic\_head\_ID, wether it’s an event or not – this propriety will be subject for further disscusion)
2. The **.entities** file: this is where all the entities found in the text are stored. Within the TSV, for each entry, we have COREF ID, used to globally identify the entity across the text in other files (.quotes, .book) what quotes are attribuited to an entity, what possesions etc. As Mattingly notes, this is one of the more challangening tasks for in NLP, therefore we should expect lower accuracy and reliability compared to the other data parameters obtained via BookNLP (around 70%). In the same file we have start and end tokens, for identifying the occourence in text of each entity mention, a prop column, where the morpohological form of the entity is mentioned (pronoun or proper noun), and a semantic category (person, facility, vehicle etc).
3. The **.quotes** file, storing all registered quotes in the text. Here, for each quote, we have its location within the text stored (quote\_start, quote\_end), the speaker location (mention\_start, mention\_end), how the speaker is reffered to (mention\_phrase) and the character ID.
4. The **.supersense** file. This file is based on the supersense taxonomy designed by Word.net. Here nouns and verbs are stored with their corresponding supersense category (The taxonomy is found in the 2.3.1 section)

One JSON file:

The **.book** file, this contains data strucured around the characters. The following JSON keys store:

* agent - actions that character does
* patient - actions done to that character
* mod - adjectives that describe them in the text
* poss - things the entity has (very broadly defined), e.g. relatives like aunt, uncle; or parts of the body, e.g. head, back, etc.
* id - their unique id (as seen above)
* g - analysis about gender pronouns used
* count - number of times the entity appears
* mentions - how the character is referenced

[Mattingly, *Introduction to BookNLP*, 2022. [booknlp.pythonhumanities.com](https://booknlp.pythonhumanities.com/booknlp.pythonhumanities.com)]

One HTML file:

The **book.html** file contains the full text and a series of highlights of data stored in the previous files, for easy user visualisation.

The other param in the model\_params variable is the spacy package, I used the “big” package, which has the best possible accuracy, at the cost of increased running time for the pipeline.

For the input and output files I used the variables of my own convention: “author” and “work”. I employ this for ease of path accesing (input\_file) and file creation (output\_directory and book\_id).

By using these variables, I run the pipeline for each text in the corpora.

**2.2.1 Supersense tags**

BookNLP’s supersense tagging functionality means assigning to each verb and noun in the input text file a tag based on the WordNet taxonomy:



[Ciaramita, Massimilano et. Altun, Yasemin, Broad-Coverage Sense Disambiguation and Information Extraction with a Supersense Sequence Tagger]

Such an exhaustive semantic taxonomy is well suited for sci-fi, a genre where authors generally employ specific terminologies for the purposes of world building (that I expect to fall mostly in the *nouns.artifact* category). Beyond proper scientific terms of various domains, repurposed to fit their particular naratological aims (ex: in “Faith of our Fathers” by Philip K. Dick, pharmaceutical vocabulary is employed, with certain substances playing a major role in the narrative), sci-fi implies terms that designate specualtive ideas and objects, products of the author’s imagination, that are more or less derived from actual science – hard sci-fi vs soft sci-fi (ex: William Gibson’s “cyberspace”, Lovecraft’s aliens and various creatures).

For my purposes, I am going to use the results offered by BookNLP to compare semantic stats between the three corpora, searching for meaningful differences/contrasts.

**2.3.2** **Obtaining the data.**

Employing Pandas, a python package designed for data manipulation and analysis within the .csv and .tsv formats, I run the following code:

import pandas as pd

df = pd.read\_csv("works/supersenseTemplate.tsv", delimiter="\t")

df1 = pd.read\_csv("works/gibson/burning\_chrome/burning\_chrome.supersense", delimiter="\t")

df2 = pd.read\_csv("works/gibson/hinterlands/hinterlands.supersense", delimiter="\t")

df3 = pd.read\_csv("works/gibson/johnny\_mnemonic/johnny\_mnemonic.supersense", delimiter="\t")

df4 = pd.read\_csv("works/gibson/neuromancer/neuromancer.supersense", delimiter="\t")

df5 = pd.read\_csv("works/gibson/new\_rose\_hotel/new\_rose\_hotel.supersense", delimiter="\t")

df1 = (df1['supersense\_category'])

df2 = (df2['supersense\_category'])

df3 = (df3['supersense\_category'])

df4 = (df4['supersense\_category'])

df5 = (df5['supersense\_category'])

df1 = df1.value\_counts()

df2 = df2.value\_counts()

df3 = df3.value\_counts()

df4 = df4.value\_counts()

df5 = df5.value\_counts()

df1.loc["noun.process"] = 0

df2.loc["noun.motive"] = 0

df3.loc["noun.motive"] = 0

df3.loc["noun.process"] = 0

df5.loc["noun.motive"] = 0

df = (df1 + df2 + df3 + df4 + df5)

df.to\_csv("works/gibson/gibson\_supersenseGlobal.tsv", sep="\t")

print (df)

I employ this procedure for each corpus. I use the read.csv Pandas’ function to initialize each .supersense file of the corpus into a Pandas dataframe. I also initialize an empty .tsv file called “supersenseTemplate” where I will store the results before writing them to a new file. I use the value\_counts() function to count each occurence of each supersense tag of each .supersense file. I have to assign 0 values to certain rows, because if a certain category is not present in all texts of the corpus, it will be missing after performing the data frames sum.

The results are then outputed in the *author\_supersenseGlobal.tsv* file. I open this file in Excel where I employ a basic math function to get percentages out of the raw values.

For further semantic analasys and comparison between the authors, I’ve also extracted unique words = words from one corpus that are not found in the other two:

import pandas as pd

df1 = pd.read\_csv("works/dick/impostor/impostor.supersense", delimiter="\t")

df2 = pd.read\_csv("works/dick/electric\_ant/electric\_ant.supersense", delimiter="\t")

df3 = pd.read\_csv("works/dick/faith\_of\_our\_fathers/faith\_of\_our\_fathers.supersense", delimiter="\t")

df4 = pd.read\_csv("works/dick/impostor/impostor.supersense", delimiter="\t")

df5 = pd.read\_csv("works/dick/minority\_report/minority\_report.supersense", delimiter="\t")

df6 = pd.read\_csv("works/gibson/burning\_chrome/burning\_chrome.supersense", delimiter="\t")

df7 = pd.read\_csv("works/gibson/hinterlands/hinterlands.supersense", delimiter="\t")

df8 = pd.read\_csv("works/gibson/johnny\_mnemonic/johnny\_mnemonic.supersense", delimiter="\t")

df9 = pd.read\_csv("works/gibson/neuromancer/neuromancer.supersense", delimiter="\t")

df10 = pd.read\_csv("works/gibson/new\_rose\_hotel/new\_rose\_hotel.supersense", delimiter="\t")

df11 = pd.read\_csv("works/lovecraft/call\_of\_cthulhu/call\_of\_cthulhu.supersense", delimiter="\t")

df12 = pd.read\_csv("works/lovecraft/at\_the\_mountains\_of\_madness/at\_the\_mountains\_of\_madness.supersense", delimiter="\t")

df13 = pd.read\_csv("works/lovecraft/the\_colour\_out\_of\_space/the\_colour\_out\_of\_space.supersense", delimiter="\t")

df14 = pd.read\_csv("works/lovecraft/the\_shadow\_over\_innsmouth/the\_shadow\_over\_innsmouth.supersense", delimiter="\t")

df15 = pd.read\_csv("works/lovecraft/the\_whisperer\_in\_darkness/the\_whisperer\_in\_darkness.supersense", delimiter="\t")

df\_dick = pd.concat([df1, df2, df3, df4, df5])

df\_gibson = pd.concat([df6, df7, df8, df9, df10])

df\_lovecraft = pd.concat([df11, df12, df13, df14, df15])

df\_not\_dick = pd.concat([df\_gibson, df\_lovecraft])

df\_not\_lovecraft = pd.concat([df\_gibson, df\_dick])

df\_not\_gibson = pd.concat([df\_lovecraft, df\_dick])

df\_merged = df\_lovecraft.merge(df\_not\_lovecraft, how="left", left\_on=["supersense\_category", "text"], right\_on=["supersense\_category", "text"], indicator=True)

df = df\_merged.query("\_merge == 'left\_only'")[["supersense\_category", "text"]]

df.to\_csv("works/lovecraft/lovecraft\_uniquesSupersense.tsv", sep="\t")

First, I read all the files of the corpora, then I concatenate them into dataframes according to author. I create “not\_auhtor” dataframes to compare against each author supersense corpus by concatenating the other two. Then, I merge the author dataframe with its opposed dataframe, allowing me to perform a query with the paramater “\_merge == ‘left\_only”, which means that I compare the left dataframe (the author) against the right dataframe (the other two authors) and delete every entry that’s mutual between the two.

I repeat this procedure for all three authors and I save the results into an “author\_uniquesSupersene.tsv” file.

Based on this, I calculate a semantic diversity percentage, by dividing unique values to the full values present in each corpus:

df\_uniques\_dick = pd.read\_csv("works/dick/dick\_uniquesSupersense.tsv", delimiter="\t")

df\_uniques\_lovecraft = pd.read\_csv("works/lovecraft/lovecraft\_uniquesSupersense.tsv", delimiter="\t")

df\_uniques\_gibson = pd.read\_csv("works/gibson/gibson\_uniquesSupersense.tsv", delimiter="\t")

print(round(len(df\_uniques\_dick)/len(df\_dick)\*100, 2))

print(round(len(df\_uniques\_lovecraft)/len(df\_lovecraft)\*100, 2))

print(round(len(df\_uniques\_gibson)/len(df\_gibson)\*100, 2))

Furthermore, I lemmatize each entry in these files and then get rid of the duplicates:

import spacy

import pandas as pd

nlp = spacy.load('en\_core\_web\_lg')

df\_uniques = pd.read\_csv("works/gibson/gibson\_uniquesSupersense.tsv", delimiter="\t")

for i, row in df\_uniques.iterrows():

    text = row['text']

    text = nlp(text)

    text = "".join([token.lemma\_ for token in text])

    row['text'] = text

df\_uniques = df\_uniques.drop\_duplicates()

df\_uniques.to\_csv("works/gibson/gibson\_uniquesSupersenseLemmas.tsv", sep="\t")

I employ these procedures for each auhtor\_uniquesSupersense.tsv file and write the new results into an author\_uniquesSupersenseLemmas.tsv file, then extract the percentages for each supersense category by author.

**2.3.3. Results and disscusion**

1.Semantic diversity

The semantic diversity percentage means the proportion of unique tagged words against all tagged words present in a corpus. The results for each author:

|  |  |
| --- | --- |
| **Author** | **Percentage** |
| Dick | 27.91% |
| Lovecraft | 38.71% |
| Gibson | 40.93% |

Ratio of unique words/total tagged words 2.1 (semantic diversity)

We observe that Lovecraft and Gibson’s values are quite close, at a ~2% difference between the two, while Philip K. Dick ranks ~10% behind them in this metric. These results seem to indicate the different writing styles and naratological approaches. Dick’s value highlights his concise, more “narrative” style. Lexical innovation and diveristy aren’t required for him, as he makes use of restrained set of key words to drive his narrative, his worldbuilding isn’t as detailed, what can be infered about the worlds he envisiones is based on the actions of his characters.

2. Total supersense percentages

The following table contains the percentages of all tagged words for each author:

|  |  |  |  |
| --- | --- | --- | --- |
| supersense\_category | lovecraft | dick | gibson |
| noun.Tops | 0.80% | 0.30% | 0.17% |
| noun.act | 3.38% | 2.64% | 2.33% |
| noun.animal | 1.39% | 1.76% | 0.61% |
| noun.artifact | 9.26% | 9.65% | 14.82% |
| noun.attribute | 3.30% | 1.65% | 1.89% |
| noun.body | 1.54% | 2.91% | 5.54% |
| noun.cognition | 4.28% | 1.87% | 1.76% |
| noun.communication | 5.29% | 4.28% | 3.69% |
| noun.event | 1.45% | 0.74% | 0.73% |
| noun.feeling | 1.29% | 0.76% | 0.44% |
| noun.food | 0.20% | 0.35% | 0.41% |
| noun.group | 2.29% | 2.30% | 1.97% |
| noun.location | 6.53% | 2.03% | 4.39% |
| noun.motive | 0.08% | 0.07% | 0.03% |
| noun.object | 3.76% | 0.68% | 1.03% |
| noun.person | 6.91% | 13.09% | 12.19% |
| noun.phenomenon | 0.74% | 0.29% | 0.52% |
| noun.plant | 0.52% | 0.13% | 0.29% |
| noun.possession | 0.16% | 0.48% | 0.50% |
| noun.process | 0.22% | 0.06% | 0.10% |
| noun.quantity | 1.11% | 0.71% | 0.85% |
| noun.relation | 0.38% | 0.26% | 0.17% |
| noun.shape | 0.35% | 0.08% | 0.38% |
| noun.state | 2.36% | 1.70% | 1.24% |
| noun.substance | 2.08% | 0.98% | 3.24% |
| noun.time | 2.89% | 2.32% | 1.79% |
| verb.body | 0.43% | 1.05% | 1.33% |
| verb.change | 3.15% | 3.29% | 2.58% |
| verb.cognition | 4.02% | 4.96% | 2.83% |
| verb.communication | 4.82% | 9.47% | 5.13% |
| verb.competition | 0.17% | 0.27% | 0.16% |
| verb.consumption | 0.37% | 0.51% | 0.55% |
| verb.contact | 2.12% | 4.26% | 5.12% |
| verb.creation | 1.03% | 0.77% | 0.80% |
| verb.emotion | 1.15% | 1.35% | 0.75% |
| verb.motion | 3.96% | 5.51% | 5.87% |
| verb.perception | 3.56% | 3.57% | 3.34% |
| verb.possession | 1.55% | 2.92% | 2.32% |
| verb.social | 2.00% | 3.06% | 1.81% |
| verb.stative | 9.07% | 6.92% | 6.19% |
| verb.weather | 0.02% | 0.01% | 0.09% |

Supersense percentages for all tagged words 2.2

3. Uniques percentages

The following table contains the percentages of only the unique words (lemmatized) for each author:

|  |  |  |  |
| --- | --- | --- | --- |
| supersense\_category | lovecraft | dick | gibson |
| noun.Tops | 0.55% | 0.18% | 0.11% |
| noun.act | 4.95% | 4.04% | 3.46% |
| noun.animal | 2.16% | 0.62% | 1.20% |
| noun.artifact | 7.61% | 11.01% | 17.85% |
| noun.attribute | 4.14% | 2.00% | 2.15% |
| noun.body | 1.32% | 1.24% | 2.50% |
| noun.cognition | 3.21% | 2.04% | 1.45% |
| noun.communication | 6.42% | 6.48% | 5.15% |
| noun.event | 2.38% | 1.02% | 1.08% |
| noun.feeling | 1.59% | 0.93% | 0.61% |
| noun.food | 0.45% | 0.49% | 0.73% |
| noun.group | 2.24% | 4.31% | 3.51% |
| noun.location | 7.64% | 3.60% | 6.41% |
| noun.motive | 0.06% | 0.09% | 0.05% |
| noun.object | 3.04% | 0.40% | 1.26% |
| noun.person | 9.63% | 11.45% | 11.83% |
| noun.phenomenon | 0.65% | 0.31% | 0.63% |
| noun.plant | 1.06% | 0.22% | 0.58% |
| noun.possession | 0.26% | 0.80% | 0.74% |
| noun.process | 0.43% | 0.04% | 0.19% |
| noun.quantity | 0.45% | 0.93% | 0.89% |
| noun.relation | 0.45% | 0.18% | 0.15% |
| noun.shape | 0.56% | 0.36% | 0.61% |
| noun.state | 2.80% | 2.22% | 1.37% |
| noun.substance | 2.02% | 1.46% | 4.06% |
| noun.time | 1.38% | 1.07% | 0.80% |
| verb.body | 0.83% | 2.13% | 1.42% |
| verb.change | 3.76% | 4.17% | 3.37% |
| verb.cognition | 3.16% | 4.31% | 1.50% |
| verb.communication | 5.12% | 6.75% | 3.11% |
| verb.competition | 0.47% | 0.71% | 0.47% |
| verb.consumption | 0.52% | 0.75% | 0.58% |
| verb.contact | 3.61% | 5.86% | 6.42% |
| verb.creation | 1.55% | 1.11% | 0.90% |
| verb.emotion | 1.76% | 1.02% | 0.62% |
| verb.motion | 3.61% | 5.99% | 5.39% |
| verb.perception | 1.37% | 2.26% | 1.72% |
| verb.possession | 0.95% | 1.42% | 1.11% |
| verb.social | 2.12% | 4.44% | 1.91% |
| verb.stative | 3.61% | 1.55% | 1.89% |
| verb.weather | 0.12% | 0.04% | 0.22% |

2.3 Percentages of supersensed tagged unique words (after lemmatization and duplicates removal)

The tagged categories in the Stanford taxonomy are numerous. I’ve highlighted the ones where significant/relevant differences are present:

*Artifacts = man-made objects*. Here, Gibson scores almost ~15%, a ~5 to 6 % increase when compared with the other two, who are very close in the 9% range (all words). For uniques, the difference increases, with Gibson reaching ~17%, and a gap of ~3% appearing between Dick (~11%) and Lovecraft (~8%).

Gibson’s shared universe is hyper-technologized and industrialized, with a constant stream of tools, gadgets, vehicles, body augmentations, and vehicles finding their way into the market (mostly the black-market). All these various artifacts are central to the narrative and overall atmoshpere, translating into the general themes: technology-inflected isolation, invasive tech, loss of boundaries between human and machine.

For Dick and Lovecraft, a significant difference appears only when comparing uniques, while in the all-words comparison, they score very close. It can be infered that man made objects populate their respective worlds simillarly from a quantitative point of view, but in Dick’s ficton, they are more varried (there are more kinds of artifacts). This is indicative of each author’s historical context. Their respective speculative visions develop the technology that was known and available to them.

In sci-fi worlds, artifacts are frequent, as many of them are the result of the technological “progress” envisioned by the author. They serve as a worldbuilding tool, as well as plot devices. However, under the umbrella of this genre, “man-made” and “progress” are relative: in Lovecraft’s universe, researches and adventurers come across vestiges left by extinct alien civilizations from time immemorial. Still, they fit into the artifacts category under the taxonomy (to the degree that BookNLP can pick them up), so for true accuracy when mining through sci-fi and fantasy texts, artifacts should be redefined as “sentient-being made objects”.

In Philip K Dick’s novel *Do androids dream of electric sheep* and the short-stories *Impostor* and *The Electric Ant*, there are android-like entities that are created specifically to resemble humans, not just phisically, where they can surpass them, but also to perform the same cognitive processes. It is natural that the pipeline assings them as persons, as this is the case at the linguistic level of the fiction. The philosopihcal implications about human nature and its relation to machines inferred by Dick are beyond BookNLP’s scope.

*The noun.body category* also highlights a signficiant difference when looking at the all words results, with ~1.5 % for Lovecraft, ~3% for Dick and 5.5% for Gibson. The differences fade out when looking at just uniques, this is natural, as body parts are overall a limited category of words. This stat infers information about the envisioned worlds, as well as the styles of the authors. Gibson’s cyberpunk universe is full of body augmentations and implants, highlighted by his cinematographical style (when new characters are introduced, it’s usually through close-ups of such glarin features, building into the theme of technological possesion). In Dick there are some simmilar instances, most notably in the beginning of *The Electric Ant*, when the main character wakes up in a hospital with a replaced hand, as he is an *electric ant*, an android whose body parts are replaceable.

*In the noun.cognition category*, Lovecraft is 2 percentages ahead of the other two, at ~4% percent in the all words statistic. The difference is dimmed when looking at the uniques, but still present, as he is still ahead by ~ 1.5%. This is indicative of Lovecraft’s overall style and focus on what the characters experience as they encounter the maddening supernatural. Also, it’s representtive of the type of characters that he prefers, who are an extension of his own persona: scholars and intelectuals who possess vast amounts of knowledge and also the vocabualry to express it, trying in vain to make sense of absurd phenomenons and beings. Example of nouns denoting cognition present only in Lovecraft’s texts: *speculation, research, contradiction, puzzlement, realisation*.

In the *noun.location* category, Lovecraft scores the highest (both for all words and uniques). His naratological appraoch involves various written reports, accounts and letters coming from many geo-locations. Gibson is the second highest, as his characters travel to various cities in their missions, there are also acounts of past events that happened around the globe for world-building purposes (especially in *Neuromancer*). Notable for Gibson is the cyberspace, a computer network where hackers transfer their consciousness. BookNLP correctly assings the cyberpsace with the *noun.location* tag, as it is a plane of existence where characters travel to, but not in a physical sense. Philip K. Dick fiction is more focused and confined, especially the short stories, his characters don’t travel as much.

In *the noun.object* (natural objects, as opposed to artifacts) category, Lovecraft is also the highest (both for all words and uniques), as his characters interact more with the natural world. This ties with the noun.location category, as he is fond of exotic environments and bioms. Examples of unique object in Lovecraft: *swamp, desert, ice barrier, valley, boulder*.

For verb labels, Dick scores higher in *cognition* and *communication*, both for all words as well as uniques. His narratological approach is very dialogue heavy (communcation). Also, he presents the monologue, internal dialogue and thought processes of his characters (cognition). This approach ties into the the themes of machines designed to resemble humans in thought and feeling.

Gibson scores highest in contact verbs, for both statistics. This is indicative of his cinematographical and action driven style, which involves movement and combat. Dick is not far behind. Lovecraft’s score is ~3 to 4 percentages, his characters are in general passive and observant, inter-character conflict and combat is rare. He is also significantly lower in motion verbs than the other two. In contrast, Lovecraft scores highest in static verbs, indicative of his characters and narrators describing and ponderion on strange sights and spaces.

Further insights about characters and “dynamism” of each corpus will be provided by the character and event stats in their respective section.

**2.3.4. Conclusions**

The semantic tags can provide insights about the overall semantic composition of narrative texts. They are plentiful in the current iteration of BookNLP, but I think that there is still room for more subspecialization in the vaster categories, such as *nouns.artifact* and *nouns.object* (for artifacts there could be subcategories such firearms, melee weapons, gadgets, clothes, household etc).

In today’s landscape of fiction, the modern reader is overwhelmed with options. Selection is often algorithm assisted. Such a semantic tagging tool could prove useful for automatic genre assigment, aswell as for large scale analsys of a genre, akin to my endeavour here. I expect that for different genres, other markers will be relevant: for romance *noun.feeling* or *verb.emotion*, for crime thrillers *verb.social* or *verb.perception*, etc.

**2.3.1 Character analysis**

Mattbew Sims and David Bamman, the developer of BookNLP, overview NLP methods developed to match sociological approaches to narative, with a focus on character networks:

“With the rise of sociological approaches to narative, work in literary criticism has increasingly turned to the ways in which authors depict social networks in their texts. This includes critical attention to both network topologies, such as understanding characters and their structural relationships with others, and information flow, such as theorizing the representation of disease and gossip. Much computational work in NLP has arisen to support the former line of research, including extracting social networks from text, predicting familial relationships, and modeling the interaction between characters. This in turn has driven work in the digital humanities examining the structure of literary works.”

[Sims, Matthew; Bamman, David, Measuring Information Propagation in Literary Social Networks].

The researchers highlight the potential uses of such tools:

“[…] understanding how the transmission of information is represented in these imagined worlds has the potential to be of great value to scholars in the humanities, since the resulting models can serve as a basis for broader insights about social strucures embeded in narratives, the role of characters based on attributes such as race, gender and informational dynamics of gossip.”

[Ibidem]

Bamman and Sims stress the inherent challange of infering nodes and edges in fictional networks. In contrast with social media, where they nested within web based meta-data, for fiction, they must be infered by the machine:

“Treating literary works themselves as networks, however, poses distinct computational challanges. While research into information propagation in social media tends to presume access to explicit networks, the character networks represented in novels are implicit.

[…]

Our goal in this work is to investigate the behaviour of information propagation in literary texts. In order to identify acts of propagation in this context, we need to determine the underlying network structure of a novel, including the nodes (by infering characters) and the edges (by inferring some interactions between them).”

[Ibidem]

BookNLP character dedicated functionalities include quote assignment that is limited to speaker identification by character ID and quote identification within the text by tokens’ ID. Currently, it doesn’t provide any social networks inferences, they could be implemented in a future iteration of the pipeline by Bamman, in line with his research.

I made use of another functionality that records character information in a *.book* extension file, as described in the documentation:

JSON file providing information about all characters mentioned more than 1 time in the book, including their proper/common/pronominal references, referential gender, actions for the which they are the agent and patient, objects they possess, and modifiers.

[BookNLP documentation]

With this, my goal is to highlight potential diffences in narrative approach and characters’ roles between the three authors.

**2.3.2 Obtaining the data**

The .book file generated by the BookNLP contains each character related data (agent and patient verbs, possesions and modifiers) in the order that they appear in text. For organizing the JSON data by character, I use Mattingly’s code. My only modification is the auhtor and work variable convention:

import json

from collections import Counter

author='lovecraft'

work='the\_whisperer\_in\_darkness'

def proc(filename):

    with open(filename) as file:

        data=json.load(file)

    return data

def get\_counter\_from\_dependency\_list(dep\_list):

    counter=Counter()

    for token in dep\_list:

        term=token["w"]

        tokenGlobalIndex=token["i"]

        counter[term]+=1

    return counter

data=proc("works" + "/" + author + "/" + work + "/" + work + ".book")

def create\_character\_data(data, printTop):

    character\_data = {}

    for character in data["characters"]:

        agentList=character["agent"]

        patientList=character["patient"]

        possList=character["poss"]

        modList=character["mod"]

        character\_id=character["id"]

        count=character["count"]

        referential\_gender\_distribution=referential\_gender\_prediction="unknown"

        if character["g"] is not None and character["g"] != "unknown":

            referential\_gender\_distribution=character["g"]["inference"]

            referential\_gender=character["g"]["argmax"]

        mentions=character["mentions"]

        proper\_mentions=mentions["proper"]

        max\_proper\_mention=""

        #Let's create some empty lists that we can append to.

        poss\_items = []

        agent\_items = []

        patient\_items = []

        mod\_items = []

        # just print out information about named characters

        if len(mentions["proper"]) > 0:

            max\_proper\_mention=mentions["proper"][0]["n"]

            for k, v in get\_counter\_from\_dependency\_list(possList).most\_common(printTop):

                poss\_items.append((v,k))

            for k, v in get\_counter\_from\_dependency\_list(agentList).most\_common(printTop):

                agent\_items.append((v,k))

            for k, v in get\_counter\_from\_dependency\_list(patientList).most\_common(printTop):

                patient\_items.append((v,k))

            for k, v in get\_counter\_from\_dependency\_list(modList).most\_common(printTop):

                mod\_items.append((v,k))

            # print(character\_id, count, max\_proper\_mention, referential\_gender)

            character\_data[character\_id] = {"id": character\_id,

                                  "count": count,

                                  "max\_proper\_mention": max\_proper\_mention,

                                  "referential\_gender": referential\_gender,

                                  "possList": poss\_items,

                                  "agentList": agent\_items,

                                  "patientList": patient\_items,

                                  "modList": mod\_items

                                 }

    return character\_data

character\_data = create\_character\_data(data, 3000)

f = open("works" + "/" + author + "/" + work + "/" + work + "Characters.json", "x")

f.write(json.dumps(character\_data))

f.close()

By this, an *workCharacters.json* file is generated for each text.

The JSON file nests each character as an object with the following proprieties: id (as inferred by the pipeline), count (how often they are mentioned), max\_proper\_mention, referential gender, possList (a list of nouns that they posses within the text), agentList (a list of verbs that they are an agent of), patientList (a list of verbs that ther are a patient of), modList (a list of adjectives and nouns that directly modify the state of character).

I encountered a problem with this particular JSON format that I fixed using regex:

import json

import re

author = 'lovecraft'

work = 'the\_whisperer\_in\_darkness'

with open('works/' + author + '/' + work + '/' + work + 'Characters.json') as f:

    data = json.load(f)

data = str(data)

data = re.sub("'+[0-9]+'+:", '', data)

data = data[1:-1]

data = "[" + data + "]"

data = re.sub("'", '"', data)

g = open('works/' + author + '/' + work + '/' + work + 'CharactersRegexFix.json', "x")

g.write(data)

g.close()

By this, an workCharactersRegexFix.json is generated for each text. Now, the JSON file can be opened for data extraction into a Pandas dataframe:

import json

import pandas as pd

author = 'gibson'

work = 'new\_rose\_hotel'

with open('works/' + author + '/' + work + '/' + work + 'CharactersRegexFix.json') as f:

    data = json.load(f)

#print(len(data))

row1 = []

row2 = []

row3 = []

row4 = []

row5 = []

row6 = []

for i in range(len(data)):

    row1.append(data[i]['max\_proper\_mention'])

for i in range(len(data)):

    row2.append(data[i]['count'])

for i in range(len(data)):

    row3.append(len(data[i]['possList']))

for i in range(len(data)):

    row4.append(len(data[i]['agentList']))

for i in range(len(data)):

    row5.append(len(data[i]['patientList']))

for i in range(len(data)):

    row6.append(len(data[i]['modList']))

df1 = pd.DataFrame(row1, columns=['Characters'])

df2 = pd.DataFrame(row2, columns = ['occourencesCount'])

df3 = pd.DataFrame(row3, columns = ['possCount'])

df4 = pd.DataFrame(row4, columns = ['agentCount'])

df5 = pd.DataFrame(row5, columns = ['patientCount'])

df6 = pd.DataFrame(row6, columns = ['modCount'])

df = pd.concat([df1,df2,df3,df4,df5,df6], axis = 1, join='inner')

df.to\_csv('works/' + author + '/' + work + '/' + work + 'CharactersStats.tsv', sep ='\t')

By this, an *workCharacterStats.tsv* is generated for each text. This file contains the count for each attribute list for each character in the respective JSON file.

**2.3.3 Results and disscusion**

Main characters

I’ve highlighted the stats of the main characters of each work. Percentages were obtained by dividing each list count to the occurence count of each character.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Work | character | occourences | possesions | agent | patient | modifiers |
| 1 | Do Androids Dream of electric sheep | Rick | 2020 | 7.77% | 19.16% | 5.40% | 1.29% |
| 2 | Electric Ant | Poole | 538 | 10.59% | 26.77% | 6.88% | 3.35% |
| 3 | Faith of our Fathers | Chien | 516 | 14.92% | 30.04% | 8.33% | 1.74% |
| 4 | Impostor | Olham | 395 | 9.87% | 32.41% | 7.59% | 2.53% |
| 5 | Minority report | Anderton | 615 | 11.54% | 34.31% | 10.08% | 2.93% |
| 6 | Dick | Avg | 4084 | 9.82% | 25.10% | 6.88% | 1.98% |

3.1 Philip K. Dick main characters stats

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Work | character | occourences | possesions | agent | patient | modifiers |
| 1 | Burning Chrome | Bobby | 171 | 20.47% | 38.60% | 3.51% | 1.75% |
| 2 | Hinterlands | Hiro | 72 | 12.50% | 44.44% | 8.33% | 6.94% |
| 3 | Johnny Mnemonic | Molly | 82 | 25.61% | 47.56% | 6.10% | 2.44% |
| 4 | Neuromancer | Case | 3595 | 9.04% | 15.80% | 4.67% | 2.09% |
| 5 | New Rose Hotel | Fox | 101 | 13.86% | 30.69% | 9.90% | 9.90% |
| 6 | Gibson | Avg | 4021 | 10.05% | 18.30% | 4.85% | 2.36% |

3.2 Gibson main characters stats

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Work | Character | occourences | possesions | agent | patient | modifiers |
| 1 | At the Mountains of Madness | Lake | 156 | 25.00% | 41.03% | 3.85% | 3.85% |
| 2 | Call of Chtulhu | Wilcox | 71 | 38.03% | 43.66% | 9.86% | 0.00% |
| 3 | The Colour of Space | Ammi | 216 | 13.43% | 43.98% | 6.94% | 3.24% |
| 4 | The Shadow over Innsmouth | Obed | 50 | 12.00% | 42.00% | 12.00% | 6.00% |
| 5 | The Whisperer in Darkness | Akeley | 11 | 27.02% | 32.58% | 10.35% | 6.06% |
| 6 | Lovecraft | Average | 504 | 23.40% | 38.25% | 8.44% | 4.50% |

3.3 Lovecraft main character stats

The issue with the main character stats occurs in Lovecraft’ corpus, as he employs only first person narrative. BookNLP currently recognizes narrator as a character entity, but it doesn’t register the verbs, possesion and modifiers associated. Another current limitation of the pipeline is that it registers only two thematic roles for the characters: agent and patient.

For Dick and Gibson, possesions and modifiers are very close, at ~10% for possesions and ~2% for modifiers. For the verbs, Dick scores notably higher in the agent category, with a ~7% increase. For the patient category, there is only a ~2% increase against Gibson. As I’ve stated, the current iteration of BookNLP provides only these 2 thematic roles. My guess is that Gibson would score significantly high in experiencer roles for his characters, in line with his cineamtic, senzorial writting style.

For Lovecraft, I chose the most mentioned characters by the narrator, they aren’t necessarily secondary characters. The agent category averages at around ~38%, while the patient at around ~8.50%. For Lovecraft in particular, this is indicative of how he depicts his characters. As I’ve previously shown in the supersense section, Lovecraft narrators are projection of his own persona. They have a tendency towards over-intellectualization, as they try to make sense of what eerie phenomena and beings. In contrast to the observant and investigative narrator, the other characters are usually active, hence they subject themselves to being agents or patient of diverse actions within the narrative.

**2.3.4 Conclusions**

Character inference is overall a hard task in NLP, espeically in narrative texts, where there is a high degree of referentiality and potential ambiguity. For large scale analasys, BookNLP’s current tools could be useful for large scale analasys of texts with narration in the third person, particularly for infering characters’ role in the text via the agent and patient verb tagging.

**2.4.1 Event detection**

In previous sections, the objects of analasys were clearly defined: semantic composition and characters. Events, however, are a more ambiguous narrative concept. In the documentation, they are defined as follows:

The event layer identifies events with asserted realis (depicted as actually taking place, with specific participants at a specific time) -- as opposed to events with other epistemic modalities (hypotheticals, future events, extradiegetic summaries by the narrator).

[BookNLP documentation]

This a simplified definition that lays out the criterion used for instructing the machine on event detection. However, an in depth definition, that lays out multiple criteria, is found in *Literary event detection,* a paper that Bamman links to in his documentation, which he co-authored alongside two other researchers:

Events remain a contested category across narrative theory, philosophy and linguistics, with definitions varying depending on discipline, application, and context. Most linguistic event classifications nevertheless trace their lineage back to Vendler (1957), who proposed four categories to distinguish the different relationships that exist between verbs and time: activities (dynamically) unfolding processes), achievments (occurances that are completed almost instantaneously), accomplishements (occurances that have some duration but also predetermined endpoint), and states (persistent conditions that span a period of time and don’t have any definite endpoint).

A simpler classification that some scholars have traced back to Aristotle (Sasse, 2002) simply distinguishes between event and states, the latter usually defined as non-dynamic situations that pertain over time. Many event annotation systems […] also treat changes of state as being events, since such changes indicate a dynamic break from prior conditions.

In our annotation approach, we include activities, achievements, accomplishments, and changes of state as being events.

[Sims, Matthew et. Park, Jong Ho et. Bamman, David, Literary event detection]

The role and scope of events within narrative is highlighted by the researchers, with reference to literary theory:

Do events determine the shape of literay naratives? This question reaches back at least as far as the 1920s, when literary theorists from the Russian Formalist school began making distinctions between syuzhet (the way in which events are presented in a narrative) and fabula (the chronological sequence of events, distinct from the way they’re represented) (Shklovsky, 1990; Propp, 2010). Even on a far more localized scale, events are often considered to play a fundamental role in how literary narratives progress. Morreti (2013), for instance, describes the inherent productivity of events in Daniel’s Defoe novel Robinson Crusoe, where one event invokes another in a chain of occurences that seem to flow in “micro-narrative sequences.” Such localized sequences in turn relate to the larger arhitecture of plot, which has its own distinct modes of organization generation (Forster, 1927; Genette, 1983; Brooks, 1992). The status of events in literature thus inevitabily engages larger questions about scale and narrative technique.

[Ibidem]

Their effort is focused on expanding NLP capacities into fiction, as in the past, event detection was mainly employed in the domain of journalism. The researchers highlight challanges inherent to narrative fiction:

The role of events in literary fiction, however, is very different from their role in fact-based reporting of events in the real world, including historical texts (Sprugnoli and Tonelli, 2017). Novels and even most short stories tend to be much longer than news articles, and tend to have more complex narrative structures both locally (individual scenes) and globally (plot) than works of non-fiction. Furthermore, literature is a creative enterprise. Journalistic discourse typically reports what actually happened in the real world and depicts definite casual chains connecting events, this causality is not hard coded into literary event sequences.

[Ibidem]

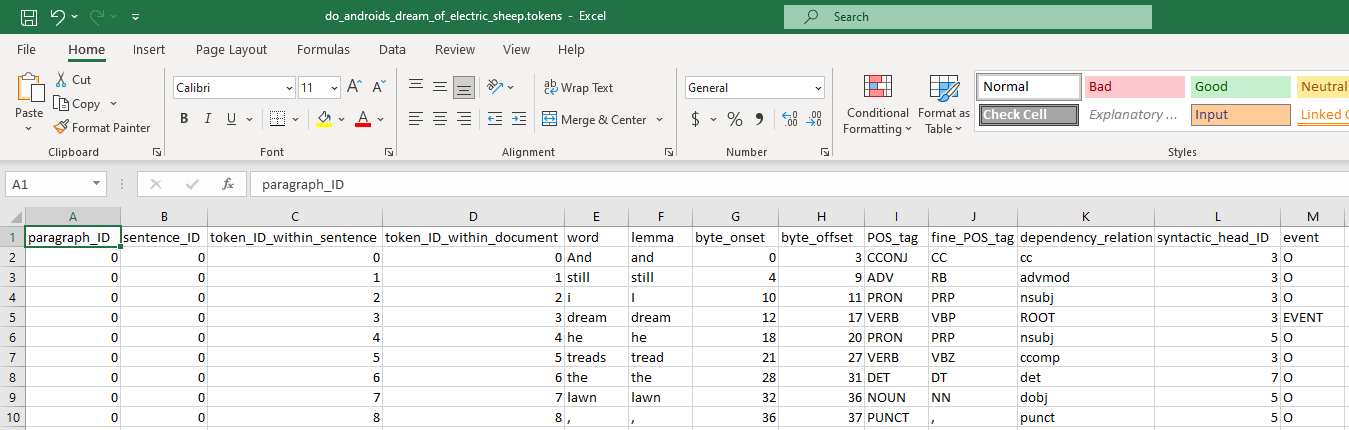
Mattingly, whose work expands on the event-tagging of BookNLP, also provides insights on how the pipeline defines and identifies events:

Event tagging is another key issue with longer documents and books. There are machine learning models that find events and you can easily cultivate a list of domain-specific events to improve a pipeline, but for BookNLP event is defined more broadly. From my experience, it is more based around key actions, rather than named events (as it is in named entity recognition). This has a tangential benefit known as triple extraction. In my opinion, it might be a bit better to view BookNLP events through this lens. Triple extraction is when we try and extract three pieces of information, such as (Actor, Action, Recipient) or (Actor, IS, Something). With these types of tuples, we can construct a knowledge tree about a corpus fairly easily. This a very challenging problem in NLP because triple extraction can be very domain-specific. BookNLP provides a great starting place for triple extraction with its events.

[Mattingly, William. *Introduction to BookNLP*, 2022. [booknlp.pythonhumanities.com](https://booknlp.pythonhumanities.com/booknlp.pythonhumanities.com).]

**2.4.2 Obtaining the data**

By deafult, BookNLP generates a *.tokens* TSVfile, which every token in the text (word or punctation mark):



I’ve extracted total number of registered events and total number of sentences for each work, and outputed the results in an *EventStats.tsv* file for each author:

import pandas as pd

import csv

author = 'gibson'

work1 = 'burning\_chrome'

work2 = 'hinterlands'

work3 = 'johnny\_mnemonic'

work4 = 'neuromancer'

work5 = 'new\_rose\_hotel'

df1 = pd.read\_csv("works/" + author + "/" + work1 + "/" + work1 + ".tokens", sep='\t', quoting=csv.QUOTE\_NONE)

df2 = pd.read\_csv("works/" + author + "/" + work2 + "/" + work2 + ".tokens", sep='\t', quoting=csv.QUOTE\_NONE)

df3 = pd.read\_csv("works/" + author + "/" + work3 + "/" + work3 + ".tokens", sep='\t', quoting=csv.QUOTE\_NONE)

df4 = pd.read\_csv("works/" + author + "/" + work4 + "/" + work4 + ".tokens", sep='\t', quoting=csv.QUOTE\_NONE)

df5 = pd.read\_csv("works/" + author + "/" + work5 + "/" + work5 + ".tokens", sep='\t', quoting=csv.QUOTE\_NONE)

rows1 = []

rows2 = []

rows3 = []

rows1.append(work1)

rows1.append(work2)

rows1.append(work3)

rows1.append(work4)

rows1.append(work5)

rows2.append(df1['event'].value\_counts()['EVENT'])

rows2.append(df2['event'].value\_counts()['EVENT'])

rows2.append(df3['event'].value\_counts()['EVENT'])

rows2.append(df4['event'].value\_counts()['EVENT'])

rows2.append(df5['event'].value\_counts()['EVENT'])

rows3.append(df1.iloc[-1][1])

rows3.append(df2.iloc[-1][1])

rows3.append(df3.iloc[-1][1])

rows3.append(df4.iloc[-1][1])

rows3.append(df5.iloc[-1][1])

df\_final1 = pd.DataFrame(rows1, columns =['Work'])

df\_final2 = pd.DataFrame(rows2, columns =['No. events'])

df\_final3 = pd.DataFrame(rows3, columns =['No. sentences'])

df\_final = pd.concat([df\_final1,df\_final2,df\_final3], axis = 1, join='inner')

df\_final.to\_csv('works/' + author + '/' + author + 'EventStats.tsv', sep ='\t')

I’ve also extracted the total number of sentences that contain at least one event (eventSents):

import pandas as pd

import csv

author = 'lovecraft'

work1 = 'at\_the\_mountains\_of\_madness'

work2 = 'call\_of\_cthulhu'

work3 = 'the\_colour\_out\_of\_space'

work4 = 'the\_shadow\_over\_innsmouth'

work5 = 'the\_whisperer\_in\_darkness'

df1 = pd.read\_csv("works/" + author + "/" + work1 + "/" + work1 + ".tokens", sep='\t', quoting=csv.QUOTE\_NONE)

df2 = pd.read\_csv("works/" + author + "/" + work2 + "/" + work2 + ".tokens", sep='\t', quoting=csv.QUOTE\_NONE)

df3 = pd.read\_csv("works/" + author + "/" + work3 + "/" + work3 + ".tokens", sep='\t', quoting=csv.QUOTE\_NONE)

df4 = pd.read\_csv("works/" + author + "/" + work4 + "/" + work4 + ".tokens", sep='\t', quoting=csv.QUOTE\_NONE)

df5 = pd.read\_csv("works/" + author + "/" + work5 + "/" + work5 + ".tokens", sep='\t', quoting=csv.QUOTE\_NONE)

sen\_ID1 = []

sen\_ID2 = []

sen\_ID3 = []

sen\_ID4 = []

sen\_ID5 = []

rows1 = []

rows2 = []

rows3 = []

rows1.append(work1)

rows1.append(work2)

rows1.append(work3)

rows1.append(work4)

rows1.append(work5)

for i in range(len(df1)):

    if (df1.iloc[i]['event'] == 'EVENT'):

        sen\_ID1.append(df1.iloc[i]['sentence\_ID'])

sen\_ID1 = list(dict.fromkeys(sen\_ID1))

sen\_ID1 = len(sen\_ID1)

for i in range(len(df2)):

    if (df2.iloc[i]['event'] == 'EVENT'):

        sen\_ID2.append(df2.iloc[i]['sentence\_ID'])

sen\_ID2 = list(dict.fromkeys(sen\_ID2))

sen\_ID2 = len(sen\_ID2)

for i in range(len(df3)):

    if (df3.iloc[i]['event'] == 'EVENT'):

        sen\_ID3.append(df3.iloc[i]['sentence\_ID'])

sen\_ID3 = list(dict.fromkeys(sen\_ID3))

sen\_ID3 = len(sen\_ID3)

for i in range(len(df4)):

    if (df4.iloc[i]['event'] == 'EVENT'):

        sen\_ID4.append(df4.iloc[i]['sentence\_ID'])

sen\_ID4 = list(dict.fromkeys(sen\_ID4))

sen\_ID4 = len(sen\_ID4)

for i in range(len(df5)):

    if (df5.iloc[i]['event'] == 'EVENT'):

        sen\_ID5.append(df5.iloc[i]['sentence\_ID'])

sen\_ID5 = list(dict.fromkeys(sen\_ID5))

sen\_ID5 = len(sen\_ID5)

rows2.append(sen\_ID1)

rows2.append(sen\_ID2)

rows2.append(sen\_ID3)

rows2.append(sen\_ID4)

rows2.append(sen\_ID5)

rows3.append(df1.iloc[-1][1])

rows3.append(df2.iloc[-1][1])

rows3.append(df3.iloc[-1][1])

rows3.append(df4.iloc[-1][1])

rows3.append(df5.iloc[-1][1])

df\_final1 = pd.DataFrame(rows1, columns =['Work'])

df\_final2 = pd.DataFrame(rows2, columns =['No. sentEvent'])

df\_final3 = pd.DataFrame(rows3, columns =['No. sentences'])

df\_final = pd.concat([df\_final1,df\_final2,df\_final3], axis = 1, join='inner')

df\_final.to\_csv('works/' + author + '/' + author + 'SentEventStats.tsv', sep ='\t')

Furthermore, I’ve created another .tsv file with all sentences, ordered by ID, and the number of events that they contain:

import pandas as pd

import csv

author = 'dick'

work = 'electric\_ant'

df = pd.read\_csv("works/" + author + "/" + work + "/" + work + ".tokens", sep='\t', quoting=csv.QUOTE\_NONE)

eventCount = 0

eventList = []

initialID = 0

for i in range(len(df)):

    if(df.iloc[i]['sentence\_ID'] == initialID):

        if (df.iloc[i]['event']) == 'EVENT':

            eventCount = eventCount + 1

    else:

        eventList.append(eventCount)

        eventCount = 0

        initialID = initialID + 1

#print (eventList)

id\_list = df['sentence\_ID'].values.tolist()

id\_list = list(dict.fromkeys(id\_list))

df\_final1 = pd.DataFrame(id\_list, columns = ['sentence\_ID'])

df\_final2 = pd.DataFrame(eventList, columns = ['eventCount'])

df\_final = pd.concat([df\_final1,df\_final2], axis = 1, join='inner')

df\_final.to\_csv('works/' + author + '/' + work + '/' + work + 'EventPerSent.tsv', sep ='\t')

This .tsv can be visualized using the Python matplotlib package, with the intent of highlighting event distribution across a text:

import pandas as pd

import matplotlib.pyplot as plt

import csv

author = 'dick'

work = 'electric\_ant'

df = pd.read\_csv("works/" + author + "/" + work + "/" + work + "EventPerSent.tsv", sep='\t', quoting=csv.QUOTE\_NONE)

df.plot(x='sentence\_ID', y='eventCount')

plt.show()

**2.4.3 Results and discussion**