**Argument**

Digital Humanties is by now an well-established field of academic research. 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 the tehnical progress that occurs within societies. 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 fundamentaly changed hows scholars structured 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 (a classic example is how ancient and medieval historians/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, I’m stating that the Digital Humanities is itself a field of subfields, an umbrella term, similarly to how the Humanities, in their traditional understanding hold the same significance. Therefore, the adjective “digital” is a consequence of the Humanities continuing to exist in the current paradigm. It isn’t, by any means, a micro-niche made up by some pretentious researchers as a result of their unconventional experimentation (although such practices have their merits and have lead to unexpected breakthroughs across all fields of research).

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 Poetica 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, occasionaly with images inserted in between. 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 phiolopshy: 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, that were 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 the controived 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 dangers of technology.

**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 far surparsses its creators. His characters deal with loss of identity, as their conciousness is being fragmented by systems outside of their understanding. The cyberpsace, a term which he coined, although occult and inaccesible for the uninitiated, still accurately portrays many aspects of today’s internet and the toll it’s taking on our lives.

**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 also comments on the pipeline’s intended use:

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.
* 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.

[Ibidem]

**1.4 Related work**

BookNLP has been employed in varoius researchers across the literary domain, which includes showcases of 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). The 4 TSV files are:

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)

The 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)]

The 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. 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, by design, implies terms that designate specualtive ideas and objects, products of the author imagination, that are more or less derived from actual science (ex: William Gibson’s “cyberspace”, Lovecraft’s aliens and various creatures). Nouns and verbs are the most productive in this regard, as they designate concepts and objects, respectevely actions associated to them that the characters perform.

For my purposes, I am going to use the results offered by BookNLP to compare semantic proportions between the three corpora, searching for meaningful distinctions

**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. 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, that being words from a 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 these 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 corpora:

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 procedure for each auhtor\_uniquesSupersense.tsv file and write the new results into an author\_uniquesSupersenseLemmas.tsv file.

**2.3.3. Results**

Citations

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.

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

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]

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 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]

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.