

Slang Analysis of Discord Data

by

Neel Gautam Bagai

Master of Education (Higher Education)

Faculty of Education

The University of Hong Kong

2025

Introduction

Human interaction has been at the heart of connection and community. It is through language and communication that individuals share ideas, build relationships and feel a sense of belonging (Warne, 2022). Over time, the ways in which humans interact have evolved with the advent of new technologies, transforming how connections are formed and maintained. In particular, the rise of online communication has revolutionized connectivity, enabling people to engage in instant messaging, voice and video calls across any distance (Li et al., 2022). Digital platforms have not only changed the speed and convenience of communication but have also given rise to entirely new linguistic trends, including the emergence of informal language, slang, and colloquialisms.

Among these platforms, Discord has emerged as a prominent space for real-time communication, offering text, voice, and video interaction within communities, referred to as "servers" (Aquino Amorim et al., 2025). Initially having been designed as a tool for gamers on different platforms to communicate with one another, Discord evolved into a platform that caters to diverse communities, from requesting relationship advice to discussing cryptocurrency investments, making it popular among various age groups, including Millennials (those born from 1981-1995), Gen Z (born from 1996-2011), and Gen Alpha (those born from 2012-2024) (Beano Brain, n.d.). Its informal and dynamic nature provides a unique environment for studying how language has evolved in digital spaces.

This study focuses on the use of informal language, including slang and colloquialisms, within Discord communities. By employing corpus-based text analysis, the goal is to find instances of such language being used, identify linguistic patterns, and explore how slang terms

interact and evolve within conversational contexts. The study also integrates entries from Urban Dictionary as a reference corpus to validate and contextualize slang usage, bridging the gap between traditional linguistic methods and contemporary online discourse.

In an era where vast amounts of information are exchanged every second in online spaces, distinguishing meaningful communication from noise has become a challenge. By analyzing informal language and its patterns in digital interactions, the results can help older demographics understand the younger generation. It can help parents to interpret the language their children use, and in an educational context, enable teachers to analyze and adapt to their students' communication styles and better cater content towards younger individuals. Ultimately, the research strives to foster clearer communication and deeper understanding within the complexities of virtual spaces, offering insights into how online communication shapes connections and linguistic evolution in the digital age.

Literature Review

The creation of a representative wordlist has long been a foundational task in corpus linguistics. Classic work by West (1953) demonstrated that the most frequent 2,000 words can account for 80% of English text, while the remaining hundreds of thousands of words comprise only 20%. This resulted in the development of the General Service List (GSL), and sixty years later, the New General Service List (NGSL) by Browne et al. (2013), which achieved 92% coverage of updated English texts with just 2,801 words. Such lists have supported second language learners of English. However, when the focus shifts to rapidly evolving online slang (on platforms such as Discord), this task becomes more complex. The project aims to create a “slang wordlist” that captures the lexical core of digital communities, with potential usefulness

for parents and educators. The ease of sending messages on Discord has resulted in large volumes of user-generated content, introducing new methodological challenges for analysis. It is therefore essential to draw on recent literature in corpus linguistics, especially studies addressing social media corpora and the challenges of analyzing slang and colloquialisms, to guide both data selection and analysis.

As the lexicon on the internet expands, the study of online slang and emergent terminology has become an integral part of society. Wilson et al. (2020) directly address this challenge by leveraging Urban Dictionary (UD), which is a vast, crowd-sourced repository of slang and non-standard language, as a primary data source for linguistic analysis. Their work is noteworthy as it validates the UD as a source of slang given that they trained, constructed, and evaluated large-scale word embeddings on UD data to capture the dynamic, context-driven meanings characteristic of online slang. This approach offered a significant improvement over traditional natural language processing resources, which are biased toward formal, standard English and thus ill-equipped to model the rapid evolution of online slang, such as that found on Discord. Through systematic evaluation using tasks such as word similarity and sentiment analysis, they demonstrated that these word embeddings outperformed baseline models, especially when handling novel or community-specific expressions. Thus, their findings validate UD as a reference corpus for studies of slang.

Wilson et al. (2020) also highlighted challenges associated with analysis of informal, crowd-sourced language data. They noted the high degree of noise, variation, and creative spelling in UD entries requires careful preprocessing such as normalization, and filtering of non-linguistic content to produce a high-quality reference corpus. Additionally, the authors

note the limitations of static embeddings, given that slang meanings are often highly context-dependent, and advocate for future research to explore contextualized language models (such as BERT) that can capture the evolving meanings of slang in online communities.

Complementing this focus on the linguistic properties of slang, Moreno-Ortiz and García-Gómez (2023) address the methodological complexities of analyzing large-scale, user-generated corpora of social media platforms. While the abundance of user-generated content theoretically helps explore presence of slang, it also introduces substantial redundancy, noise, and computational challenges, which would be present in Discord data given the conversational nature of the platform. The authors systematically compare sampling methodologies, finding that proportional-to-size sampling (PPS) is more effective than random sampling for social media datasets. PPS preserves the distributional characteristics of the original corpus while significantly reducing computational resources required for evaluation. Their results reveal that, even with a reduction in sample size, PPS produces keyword extraction results comparable to larger samples, challenging the assumption that more data necessarily yields better insights. This is valuable to know if the analysis of the Discord data takes a long time to run. In said case, strategically sampled subsets of the data can be used to create representative slang wordlists in a more efficient manner.

In evaluating keyword extraction techniques, Moreno-Ortiz and García-Gómez (2023) contrast traditional frequency-based methods, which rely on reference corpora, with graph-based approaches such as TextRank. While frequency-based methods can identify high-frequency or domain-specific terms, they can be distorted by the choice of reference corpus and are vulnerable to extracting function words or irrelevant tokens, especially in noisy datasets.

Graph-based methods leverage word co-occurrence and relational patterns, which are shown to be more effective at detecting emergent and colloquial expressions that define Discord slang but may not appear frequently. The study also emphasizes the importance of transparent preprocessing such as removing irrelevant bot messages or hyperlinks to help enhance the quality of extracted keywords.

Together, the insights of Wilson et al. (2020) and Moreno-Ortiz & García-Gómez (2023) offer a valuable methodological foundation for compiling a representative slang wordlist from Discord. By validating Urban Dictionary as a reference and using various methods for sampling and extraction, it can help in creating a wordlist that captures the slang of Discord messages.

Research Questions

1. To what extent is informal language, including slang and colloquialisms, present in Discord communities?
2. What are the most used slang terms and informal expressions on Discord, and how can they be systematically documented to assist parents and educators in understanding and engaging with younger generations?

Methodology

This study employs a corpus linguistics approach to analyze the prevalence and nature of slang and informal language in Discord communities, utilizing the insights gained from the literature review. The first stage involved constructing a reference slang corpus by utilizing the available sitemap of UD to scrape one-word slang terms. With the help of OpenAI's GPT-4o

and GPT 4.1, Python scripts were developed to access and parse the platform's one-word slangs from A-Z. This approach, guided by the methodology of Wilson et al. (2020), ensured the creation of a relevant slang lexicon while minimizing ambiguity and noise. The resulting dataset provided a useful reference against which Discord data could be compared.

The next stage centered on the preprocessing of Discord chat data that was downloaded for free from Kaggle (Fan, 2021). Given Discord's terms of service (Discord Terms of Service, 2024), scraping is not allowed and the Kaggle dataset was built through crowdsourcing, where discord server moderators (administrator) submitted logs to Fan (2021), the curator of the dataset. Following the advice from the scholarly literature available, the data underwent systematic cleaning using Python. This process included tokenization, lemmatization, and the removal of stop words, URLs, emojis, and non-linguistic artifacts such as bot messages and URLs. All messages were normalized to lowercase for ease, resulting in a set of 736 clean, lemmatized text files ready for analysis. Each text file represented messages from a different server. This step was essential for reducing noise and ensuring the reliability of subsequent frequency-based analyses, as supported by the findings of Moreno-Ortiz and García-Gómez (2023).

Subsequently, the cleaned Discord dataset was analyzed using AntConc, with the UD list serving as the reference corpus. Keyword extraction was performed to identify words that were significantly more frequent in the Discord corpus compared to the reference. To enhance the quality of the findings, a minimum frequency threshold of 200 occurrences and a range requirement of presence in at least 50 out of 736 Discord servers (text files) were imposed. This ensured that only widely used terms were retained. The resulting keyword list,

highlighting prominent slang and informal expressions, was exported as a comma separated values (csv) file for further refinement.

The next stage involved the lexical categorization and filtering of keywords. Using the free WordNet dictionary data (Provalis Research, 2025), the .CAT files were converted to a compatible .txt format using Notepad++. The dictionary WordNet data was combined in Python, with duplicates being removed and classifications (noun, adverb etc.) for each word extracted. Using massive WordNet list with their classification categories, along with subcategories (noun.communication, adverb.all), it served as a reference point for analyzing the keyword list generated by AntConc based on the Discord Data. Only those keywords classified as noun.communication (communication-based nouns) and those with no category were retained to create the final list of Discord slang, with all stop words having been removed using the Natural Language Tool Kit (NLTK) in Python. This process yielded a refined, actionable list of communication-related slang terms prevalent in Discord communities that contained 2721 words compared to the 6266 words in the 'keywordlist.csv' file exported by AntConc of words present in the UD reference corpus that were present in the Discord Data and appeared over 200 times and in at least 50 or more .txt files.

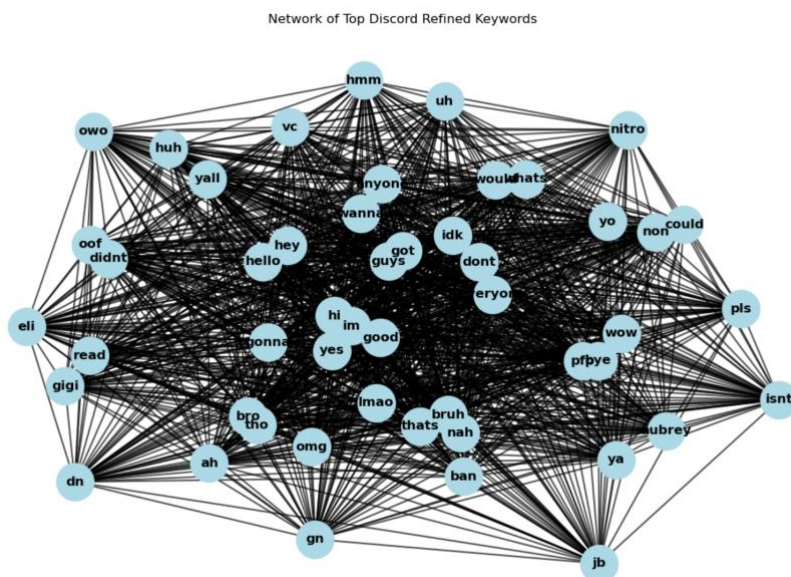
To assess the prominence of each slang term within the Discord corpus, a co-occurrence network was constructed from the refined keyword list containing 2721 words, and PageRank centrality scores were computed based on the cleaned Discord Data. PageRank identified the terms that were both frequent and contextually connected to other important terms, allowing for the ranking of slang by their influence within the Discord data. This graph-based method yielded a more meaningful measure of importance of slang than raw frequency alone, making

it easy to see the slang in order of prominence within Discord communities. To obtain a refined list of slang, the results from the graph-based keyword extraction were manually dealt with to create the final list of slang words. For any unknown slang, key word in context (KWIC) in AntConc was used to gauge the meaning of the word before being cross checked online on UD.

Detailed information regarding the methodology, such as screenshots of the code and outputs can be found in Appendix B. All files utilized for data analysis can be found at: https://github.com/neelbagai2209/MEDD8925_Discord_Textual_Analysis

Presentation of Findings

Figure 1. Co-occurrence network of Discord Slang



As can be seen in Figure 1 above, the most prominent slang terms are shown. However, the formatting is not very good, so the list sorted using PageRank score will be used instead.

Table 1. Preview of Keywords with Page Rank Score

keyword	pagerank_score
im	0.00878611
yes	0.00468714
hi	0.00446548
good	0.00430634
dont	0.00375847
got	0.00329343
hey	0.0024589
hello	0.00197614
guys	0.00196421
thats	0.00193486
wanna	0.00192757
lmao	0.00168842
gonna	0.00160969
bruh	0.00134318
would	0.00134077
idk	0.00133371
ban	0.0012223
everyone	0.00121575
anyone	0.00114365
bro	0.00111543
jb	0.00110657
omg	0.00110398

As can be seen above, the raw output from the Python file included words that were slang and used in other contexts. There were 2722 entries and only the ones with a decimal PageRank score were retained. Any word which had a PageRank score that was small and in scientific notation, was deleted due to its irrelevance compared to more widely used words. It resulted in the following list below, containing 234 words, which can be found in Table 2.

Table 2. Preview of Sorted Slang with Page Rank Scores

keyword	pagerank score
im	0.008786107
wanna	0.001927572
lmao	0.001688417
gonna	0.001609687
bruh	0.001343182
idk	0.001333712
bro	0.001115431
jb	0.001106567
omg	0.001103983
nah	0.00097847
yo	0.000955157
tho	0.00094719
vc	0.000939111
pfp	0.000902513
aubrey	0.000768252
ya	0.000739833
pls	0.000731336
eli	0.00071537
gn	0.000709785
gigi	0.000684918
yall	0.000682658

The above is just a preview of the data, showing 21 out of the 234 words. There are the most common slangs like ‘im’ which is short for ‘I am’ and ‘yall’ which is short for ‘you all’ and even ‘omg’ which is short for ‘oh my god.’ However, there are some slang unknown to myself, such as ‘aubrey,’ ‘eli’ and ‘gigi,’ for which KWIC analysis was used to generate concordance lines to gauge the meaning, before checking it online on UD.

Figure 2. Preview of KWIC analysis in AntConc of the word ‘aubrey’ in the Discord Data

KWIC and concordance lines for ‘aubrey’ show that it is simply someone’s name so it can be omitted from the final list with meaning that can be found in Appendix A.

Figure 3. Preview of KWIC analysis in AntConc of the word ‘eli’ in the Discord Data

Based on figure 3, it also seems that the word ‘eli’ is community specific and is not some slang that is widely used. It is noteworthy that it is used with the word ‘hug’ in all entries.

Figure 4. Preview of KWIC analysis in AntConc of the word ‘gigi in the Discord Data

Line	Keyline	Text
1	21486717	aye gigi gigi hi yolo yolo bawedigi gigi hi gigi
2	21486717	we h a t e e you're the best subject
3	21486717	deletedchannel, i don't know neither (what? hi yay yay)
4	21486717	gigi you made pil look under her bed
5	21486717	we soon coo-hi-in at while gigi? lmao
6	21486717	orig: kias kias gigi can i be the referee i off
7	21486717	mua eat a kias pat onyug hi sub-wisdom? ping
8	21486717	pls aight gigi she gotta a pretty face, is o
9	21486717	yes that's i good luck uni? gigi i am racist
10	21486717	hennemaneh gigi alighty right gigi on are you baby
11	21486717	no life and is here 24 gigi kias hug, hug rhuggs
12	21486717	no offense im king of shag gigi cat lady deaf
13	21486717	go downwax gigi like 7 years ago hi kias kias gassy
14	21486717	some snakes i okay gigi i don't figns are cute
15	21486717	afica for assest in pls of my eyes hi gigi
16	21486717	fraser pack kias gigi fraser pack kias in the best
17	21486717	hennemaneh my hand is bigger than my memos not everyone
18	21486717	gigi like the police? gigi by goddness you're gass
19	21486717	what are you gigi goes like the back on maderpissan
20	21486717	each dypse and that too piece of cake to herself
21	21486717	gigi dypse i'm i'm gigi i'm gonna cry
22	21486717	for time into gag gigi like should i sleep? hey
23	21486717	hennemaneh do you know any hi is longer nice
24	21486717	have 1 brother gigi so i am impressed by that dery
25	21486717	don't show gigi when i'm gigi i'm coss
26	21486717	your when you don't watch but everyone who does
27	21486717	gigi he called you hot gas what is going on
28	21486717	gigi hi i'm i can forget a computer
29	21486717	hennemaneh hi is a coss, i am a winner too
30	21486717	don't 100% i don't instead of speedy gigi kias gas fish
31	21486717	hi gigi hi hi hi hi hi

Based on figure 4, it appears that ‘gigi’ is referring to someone’s name and is not some slang being used, similar to ‘aubrey’ and ‘eli.’ These words were omitted from the final slang list that serves as the deliverable for teachers and parents. Repeating this for all the unfamiliar slang was an unexpected part of the research, but necessary to ensure good quality of findings.

Using the final list of slang, which contained 193 terms, a simple frequency analysis in AntConc showed the sum of frequencies for the 193 terms to be 8,386,086. Given the cleaned Discord Data is 214,864,717, slang words make up 3.90% of the total corpus.

Conclusion and Limitations

This study aimed to examine the extent and nature of informal language within Discord communities. The results proved that slang is a notable feature of communication on Discord, accounting for 3.9% of the total words in the analyzed corpus. The most prominent slang terms were “im,” “wanna,” “lmao,” and “bruh,” offering a valuable resource for parents and

educators seeking to better understand and engage with younger generations' online discourse which can be found in Appendix A. Some of the slang was quite gaming specific, but this is to be expected on a platform such as Discord, although this trend is unlikely to be present in analysis of other social media platforms.

However, several limitations must be acknowledged. Firstly, data access was restricted due to legal and ethical considerations. Scraping Discord directly was not permitted, so the study relied on a pre-collected, publicly available dataset. Secondly, the analysis focused exclusively on one-word slang terms, omitting multi-word expressions like “spill the tea,” which are also prevalent in digital communication. Thirdly, while a co-occurrence graph was generated, its complexity limited its interpretability, though PageRank analysis proved useful. Lastly, manual review and refinement of the slang list were required to remove irrelevant or context-specific terms such as names. While this was feasible for the current dataset, scalability may pose challenges for larger studies and will require more automated, robust methods in the future.

Word count (excluding in-text citations, headings, figure/table labels, and tables): 2,472

References

- Aquino Amorim, Y., Bento, P., Buzelin, A., Malaquias, S., Santana, C., Estanislau, V., Dutenhofner, P., Evangelista, G., Porfirio, L., Grossi, C., Rigueira, P., Almeida, V., Pappa, G., & Meira Jr, W. (2025). *Discord Unveiled: A Comprehensive Dataset of Public Communication (2015-2024)*. <https://doi.org/10.48550/arXiv.2502.00627>
- Beano Brain. (n.d.). *Discord: Redefining social media for Gen Alpha*. Retrieved March 1, 2025, from <https://insights.beanobrain.com/discord-redefining-social-media-for-gen-alpha>
- Browne, C., Culligan, B., & Phillips, J. (2013). *The New General Service List*. <http://www.newgeneralservicelist.org>
- Discord Terms of Service*. (2024, March 15). <https://discord.com/terms>
- Fan, J. (2021). *Discord-Data*. <https://www.kaggle.com/datasets/jef1056/discord-data>
- Li, C., Ning, G., Xia, Y., Guo, K., & Liu, Q. (2022). Does the Internet Bring People Closer Together or Further Apart? The Impact of Internet Usage on Interpersonal Communications. *Behavioral Sciences*, 12, 425. <https://doi.org/10.3390/bs12110425>
- Moreno-Ortiz, A., & García-Gámez, M. (2023). Strategies for the Analysis of Large Social Media Corpora: Sampling and Keyword Extraction Methods. *Corpus Pragmatics*, 7(3), 241–265. <https://doi.org/10.1007/s41701-023-00143-0>
- Provalis Research. (2025). *Wordnet based categorization dictionary*. <https://provalisresearch.com/products/content-analysis-software/wordstat-dictionary/wordnet-based-categorization-dictionary/>
- Urban Dictionary*. (2025). Urban Dictionary. <https://www.urbandictionary.com/>
- Warne, J. (2022, November 8). *The psychology of community: Why human connection makes CLG work*. Community-Led Alliance. <https://www.communityledgrowth.com/the-psychology-of-community-why-human-connection-makes-clg-work/>

West, M. (1953). *A General Service List of English Words*. Longman.

Wilson, S., Magdy, W., McGillivray, B., Garimella, K., & Tyson, G. (2020). *Urban*

Dictionary Embeddings for Slang NLP Applications (N. Calzolari, F. Béchet, P.

Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J.

Mariani, H. Mazo, A. Moreno, J. Odijk, & S. Piperidis, Eds.; pp. 4764–4773).

European Language Resources Association. <https://aclanthology.org/2020.lrec-1.586/>

Appendices

Appendix A. Slang terms along with their meanings (most definitions self-provided)

Slang	Meaning
im	I am
wanna	Want to
lmao	laughing my ass off
gonna	going to
bruh	brother
idk	I don't know
bro	brother
omg	Oh my god
nah	No
yo	Informal greeting
tho	Though
vc	Voice chat / video call
pfp	Profile picture
ya	Yes
pls	Please
gn	Good night
yall	You all
huh	Confused expression
ah	Realization of something
uh	Filler word for thinking
hmm	Filler word
oof	Expression when something bad happens
nitro	Discord premium
tf	The fuck
ye	Yes
imma	I'm going to
yn	You know
cuz	Because
ive	I have
afk	Away from keyboard
ik	I know
gotta	Got to
uwu	Expression representing a cute face*
simp	Simpleton
ty	Thank you
lil	Little
btw	By the way
hmmm	Expression when thinking
ppl	People
hiii	Informal greeting

dms	Direct messages
ima	I'm going to
woah	Expression when surprised
lvl	Level
eh	Just saying 'eh'
hai	Informal greeting
gg	Good game
skrrt	The sound made by wheels of a vehicle screeching*
um	For when you don't know what to say
wat	What
hii	Informal greeting
nvm	Nevermind
smh	Shake my head (in disappointment)
ily	I love you
oop	Expression when you make a mistake
jk	Just kidding
ooo	Expression when something interesting happens
ln	Last night
wdym	What do you mean?
yay	Expression when you're excited and happy
cry	Refers to either crying from sadness or from laughter
aight	Alright
cya	See you
mhm	Another way of saying 'hmm'
died	Not living OR dying from laughter
nope	No
minecraft	A popular game
oi	An informal way to call someone. Oi <insert name>
lemme	Let me
fn	For now
lmfao	Laughing my fucking ass off
oo	Another way of saying 'ooo'
wha	Another way of saying what
ikr	I know right
pp	Either refers to profile picture or to someone's penis
na	A way of saying no or nah
ded	Another way of saying 'dead,' related to dying from something funny
nooo	No
wut	What
tbh	To be honest
ow	Expression for when someone is in pain, similar to 'ouch'
pog	Abbreviation for 'play of the game,' used when something great is done
wtf	What the fuck
roblox	A popular video game
yep	Yes
lol	Laughing out loud

uhh	Filler word for when you are unsure
woke	Term used for people that are extremely politically correct
boi	Boy
pics	Pictures
pinged	The past tense of someone having sent a ping
uhhh	Filler word for when you are unsure, like 'uhh'
ohhh	Another way of saying 'oh'
qt	A way of saying cutie
chan	Someone's name or referring to 4chan (numbers were removed from data)
ayo	An expression when something unexpected happens. Contraction of 'Ey yo'
gf	Girlfriend
welp	Informal expression indicating sadness or disappointment
nsfw	Abbreviation for 'not safe for work'
hbu	Abbreviation for 'how about you?'
weeb	Derogatory word to describe those into Japanese culture such as anime
xd	An expression for a laughing face and an alternative to 'lol'
nani	Japanese for 'what' *
hehe	Another way of saying 'haha'
emotes	Refers to the dances present in the fortnite game used to taunt opponents
ofc	Abbreviation for 'of course'
wbu	Abbreviation for 'what about you'
hihi	Another way of saying 'hi'
congrats	Another way of saying 'congratulations'
nom	Refers to the action of eating something
aww	Expression for when someone finds something cute
abt	Slang for 'about'
heya	Hey you
boop	Just an expression for affectionately poking someone
toon	Refers to a character in a role-playing game*
emoji	Refers to the use of emojis
aw	Another way of saying 'aww'
ngl	Abbreviation of 'not going to lie'
cutie	Refers to someone that is cute
oki	Another way of saying 'okay'
dw	Abbreviation for 'don't worry'
ohh	Another way of saying 'oh'
gon	Going to
irl	In real life
fav	Favourite
gimme	Give me
fortnite	A popular video game
legit	Expression used when something is real
ahh	Either an expression of realization or a better way of saying 'ass'
wot	What
doin	Doing
tryna	Trying to

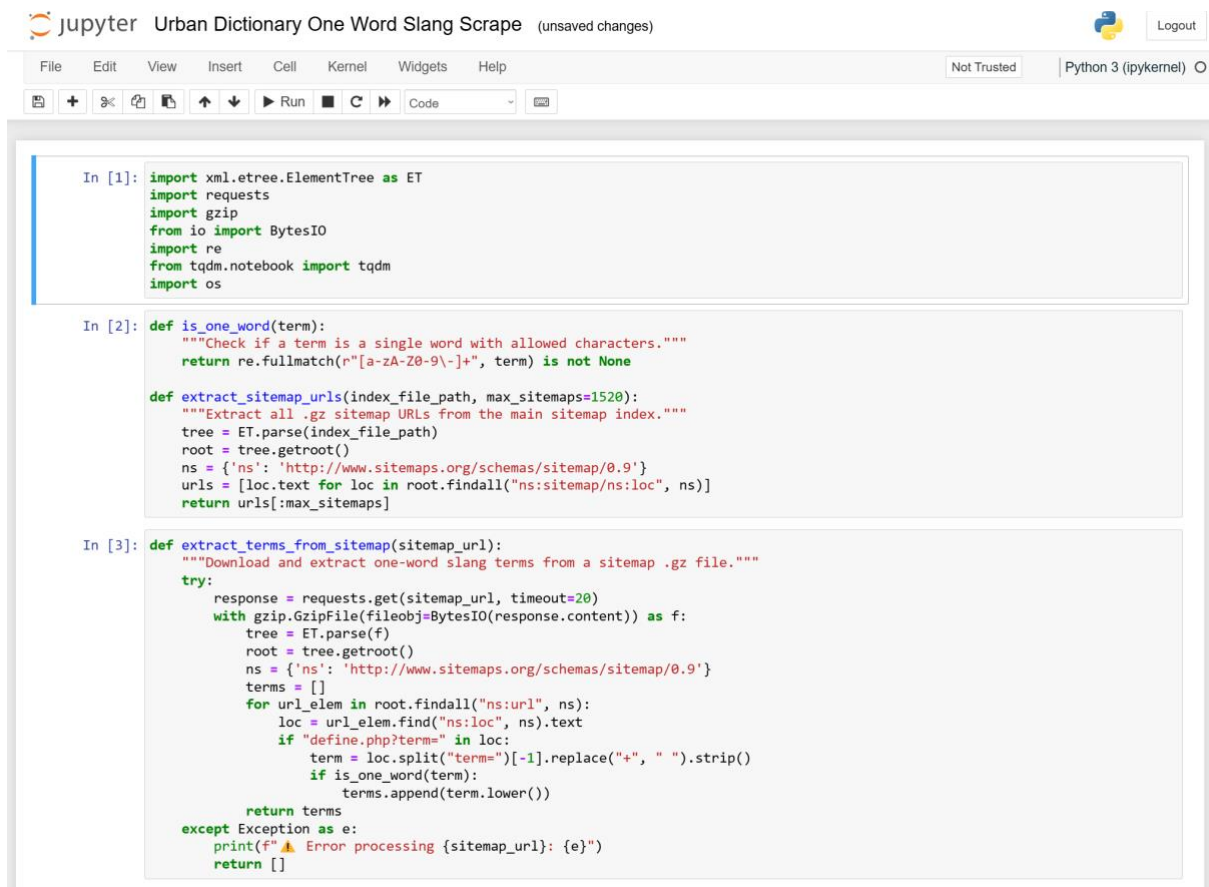
meh	Expression for when one has an indifferent reaction towards something
gif	Refers to .gif files which are short videos that play on loop
rlly	Really
ooh	Another way of saying 'oh'
thx	Another way of saying 'thanks'
dang	Expression for when something happens. An alternative to 'damn'
ugh	Expression for when someone is frustrated / angry with something
heyo	A slang for 'hey yo' or alternative to saying 'whatsup'
gtg	Got to go
aye	Another way of saying 'ayy'
yeet	An expression to indicate enthusiasm OR to throw / chuck something
brb	Be right back
gud	Good
af	Abbreviation for 'as fuck'
ahhh	Another way of saying 'ahh'
manga	Refers to Japanese comics
yup	Yes
ight	Another way of saying 'aight' which is short for alright
senpai	Refers to a master of something
alr	A shorter way of saying 'already'
waifu	Refers to a wife or romantic partner
hentai	Japanese pornography with animated characters
osu	A popular Japanese video game
sooo	Another way of saying 'so'
kek	An alternative to 'lol' or 'haha' to indicate laughter in relation to something funny
memes	Refers to 'memes' (pictures with funny captions)
ay	Another way of saying 'ayy' or 'ey'
feels	An overwhelming emotional reaction
wassup	Whatsup
bday	Birthday
bf	Boyfriend
si	Spanish for 'yes'
homie	A good friend and another way of saying 'home boy'
yooo	A longer way of saying 'Yo,' an informal greeting
okie	Another way of saying 'okay'
bai	Another way of saying 'bye'
mic	Refers to someone's microphone, particularly related to voice calls
spotify	A popular music streaming platform
ps	An additional thought OR short for 'photoshop' OR 'playstation'
dunno	Don't know
yikes	Expression for when something bad happens
ello	Another way of saying 'hello'
min	Refers to 'minute' and in context one would say 'wait a min'
mi	Spanish for 'me'
chu	Another way of saying 'you' and in context, 'wi chu' (with you)
lmaoo	Another way of saying 'lmao' (laughing my ass off)

hiya	Hey you
uhm	Another way of saying 'umm' (a filler word)
soo	Another way of saying 'so'
smol	A slang for the word 'small'
lmaooo	Another way of saying 'lmao' (laughing my ass off)
plz	Another way of saying 'pls' (please)
stfu	Shut the fuck up
rn	Right now
oops	Expression for when one makes a mistake
idc	I don't care
umm	A filler word for when one is thinking
smth	Something
ez	Easy
noni	Another way of saying 'nani' (Japanese for 'what')*
heh	An expression for when one is confused, similar to 'huh'

* = Meaning taken from (Urban Dictionary, 2025)

Appendix B. Screenshots of Data Analysis

Part 1- Urban Dictionary Scrapping



The screenshot displays a Jupyter Notebook titled "Urban Dictionary One Word Slang Scrape" with "(unsaved changes)" indicated. The interface includes a top bar with the Jupyter logo, a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a status bar (Not Trusted, Python 3 (ipykernel)), and a toolbar with icons for file operations, running, and code execution. The notebook contains three code cells:

```
In [1]: import xml.etree.ElementTree as ET
import requests
import gzip
from io import BytesIO
import re
from tqdm.notebook import tqdm
import os

In [2]: def is_one_word(term):
    """Check if a term is a single word with allowed characters."""
    return re.fullmatch("[a-zA-Z0-9\~\-\+]", term) is not None

    def extract_sitemap_urls(index_file_path, max_sitemaps=1520):
        """Extract all .gz sitemap URLs from the main sitemap index."""
        tree = ET.parse(index_file_path)
        root = tree.getroot()
        ns = {'ns': 'http://www.sitemaps.org/schemas/sitemap/0.9'}
        urls = [loc.text for loc in root.findall("ns:sitemap/ns:loc", ns)]
        return urls[:max_sitemaps]

In [3]: def extract_terms_from_sitemap(sitemap_url):
    """Download and extract one-word slang terms from a sitemap .gz file."""
    try:
        response = requests.get(sitemap_url, timeout=20)
        with gzip.GzipFile(fileobj=BytesIO(response.content)) as f:
            tree = ET.parse(f)
            root = tree.getroot()
            ns = {'ns': 'http://www.sitemaps.org/schemas/sitemap/0.9'}
            terms = []
            for url_elem in root.findall("ns:url", ns):
                loc = url_elem.find("ns:loc", ns).text
                if "define.php?term=" in loc:
                    term = loc.split("term=")[-1].replace("+", " ").strip()
                    if is_one_word(term):
                        terms.append(term.lower())
            return terms
    except Exception as e:
        print(f"⚠️ Error processing {sitemap_url}: {e}")
        return []
```

```
In [4]: # === CONFIGURATION ===
SITEMAP_INDEX_FILE = r"C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\Final Project\urbandict_sitemap_gz.xml"
MAX_SITEMAPS = 1520 # You can reduce this temporarily for testing
OUTPUT_FILE = "urban_dictionary_one_word_slangs.txt"

# === MAIN PROCESS ===

sitemap_urls = extract_sitemap_urls(SITEMAP_INDEX_FILE, max_sitemaps=MAX_SITEMAPS)
print(f"🔍 Found {len(sitemap_urls)} sitemap files.")

all_terms = set()

for sitemap_url in tqdm(sitemap_urls, desc="🔄 Processing sitemaps"):
    terms = extract_terms_from_sitemap(sitemap_url)
    all_terms.update(terms)

print(f"\n✅ Extracted {len(all_terms):,} unique one-word slang terms.")

🔍 Found 1520 sitemap files.
🔄 Processing sitemaps: 0%|          | 0/1520 [00:00<?, ?it/s]

✅ Extracted 1,636,457 unique one-word slang terms.

In [6]: # 💡 Filter out any terms that contain non-alphabetic characters
filtered_terms = {term for term in all_terms if term.isalpha()}

print(f"🔪 Filtered terms: {len(filtered_terms):,} remain after removing non-alphabetic ones.")

🔪 Filtered terms: 1,540,197 remain after removing non-alphabetic ones.

In [7]: with open(OUTPUT_FILE, "w", encoding="utf-8") as f:
        for term in sorted(filtered_terms):
            f.write(term + "\n")

print(f"💾 Saved one-word slang terms to: {os.path.abspath(OUTPUT_FILE)}")

💾 Saved one-word slang terms to: C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\Final Project\urban_dictionary_one_word_slangs.txt
```

Part 2 - Discord Data Cleanup


Jupyter
Discord Data Cleanup
Last Checkpoint: 03/27/2025 (autosaved)

Logout

File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)











Code

```
In [1]: import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import emoji # Install with `!pip install emoji`

# Download necessary NLTK resources (run this once)
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')

# Initialize tools
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words("english"))

[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\neelb\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\neelb\AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data] C:\Users\neelb\AppData\Roaming\nltk_data...
[nltk_data] Package wordnet is already up-to-date!
```

```
In [2]: def remove_usernames(text):
        """
        Removes usernames that appear at the start of the line or are followed by a colon.
        Replaces the username with a space to avoid concatenation issues.
        """
        # Replace username pattern with a space
        text = re.sub(r"[a-zA-Z0-9_]{3,}:", " ", text)
        return text
```

```
In [3]: def remove_bot_commands(text):
        """
        Removes bot commands like '!command' or '?command' and @mentions.
        """
        text = re.sub(r'\?\\w+\\s?@\\w+', '', text) # Remove bot commands Like "?warn @user"
        text = re.sub(r'!\\w+\\s?@\\w+', '', text) # Remove bot commands Like "!rank @user"
        text = re.sub(r'@\\w+', '', text) # Remove @mentions
        return text
```

```
In [4]: def format_text(text):
        """
        Formats text by:
        - Replacing multiple newlines with single spaces.
        - Ensuring spaces exist around punctuation.
        - Correcting common phrasing issues.
        """
        # Replace multiple newlines or mixed spaces/newLines with a single space
        text = re.sub(r'\\s*\\n\\s*', ' ', text)

        # Add a space after punctuation if it's missing
        text = re.sub(r'(?<[.,!?])(?!=\\S)', ' ', text)

        # Correct specific known concatenation patterns
        text = re.sub(r'\\bitnthat's\\b', "it that's", text, flags=re.IGNORECASE) # Correct "itnthat's"
        text = re.sub(r'\\btherewego\\b', "there we go", text, flags=re.IGNORECASE) # Correct "therewego"
        text = re.sub(r'\\bgonlmao\\b', "gon lmao", text, flags=re.IGNORECASE) # Correct "gonlmao"

        # Ensure proper spacing between words
        text = re.sub(r'(?<=[a-zA-Z])(?=[A-Z])', ' ', text) # Split camel case, if any
        text = re.sub(r'(?<=[a-zA-Z])(?=[0-9])', ' ', text) # Split words from numbers, if concatenated

        # Strip extra spaces
        text = " ".join(text.split())

        return text
```

```
In [5]: def clean_text(text):
        """
        Cleans and preprocesses text:
        - Removes usernames, bot commands, and emojis.
        - Retains meaningful structure without removing critical stopwords or context.
        """
        text = remove_usernames(text) # Remove usernames
        text = remove_bot_commands(text) # Remove bot commands
        text = format_text(text) # Properly handle newlines and spacing
        text = emoji.replace_emoji(text, replace="") # Remove emojis
        text = re.sub(r"http\\S+|www\\S+", "", text) # Remove URLs
        text = re.sub(r"[^a-zA-Z0-9\\s,!?]", "", text) # Retain alphanumeric characters and basic punctuation
        text = text.lower().strip() # Convert to lowercase

        return text
```

```
In [6]: def clean_text_file(input_file, output_file):
        """
        Reads a text file, cleans its contents, and writes the cleaned data to a new file.
        """
        cleaned_lines = []

        with open(input_file, 'r', encoding='utf-8') as file:
            for line in file:
                cleaned_line = clean_text(line) # Apply the cleaning pipeline
                cleaned_lines.append(cleaned_line.strip())

        # Write the cleaned data to the output file
        with open(output_file, 'w', encoding='utf-8') as file:
            file.write("\\n".join(cleaned_lines))
```

```
In [7]: import os

def process_all_files(input_folder, output_folder):
    """
    Processes all text files in the input folder, cleans them, and saves the results in the output folder.
    """
    # Create the output folder if it doesn't exist
    os.makedirs(output_folder, exist_ok=True)

    # Process each text file in the input folder
    for filename in os.listdir(input_folder):
        if filename.endswith(".txt"):
            input_file_path = os.path.join(input_folder, filename)
            output_file_name = os.path.splitext(filename)[0] + "_Cleaned.txt"
            output_file_path = os.path.join(output_folder, output_file_name)

            clean_text_file(input_file_path, output_file_path)
            print(f"✅ Processed and cleaned: {filename}")

    print(f"✅ All files cleaned and saved in {output_folder}")
```

```

In [8]: # Example problematic text
example_text = "middle eastern: not, itnthats how it is therewego lmao gonlmao i can almost guarantee itnthats true"

# Clean the text
cleaned_text = clean_text(example_text)
print("Original Text:", example_text)
print("Cleaned Text:", cleaned_text)

Original Text: middle eastern: not, itnthats how it is therewego lmao gonlmao i can almost guarantee itnthats true
Cleaned Text: not, it that's how it is there we go lmao gon lmao i can almost guarantee it that's true

In [9]: # Define input and output folders
input_folder = r"C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\Discord Data v3 Antispa
output_folder = r"C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\Cleaned Discord Data"

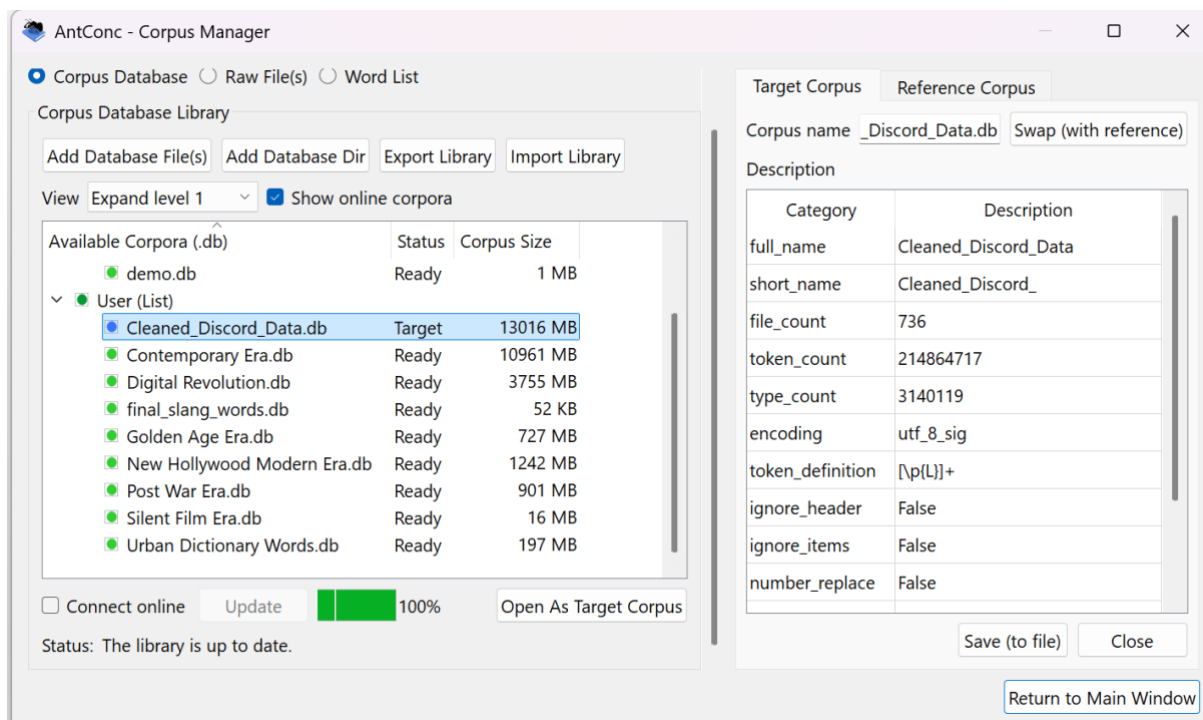
# Process all files in the folder
process_all_files(input_folder, output_folder)

[792868077413859348].txt
[794540887380066304].txt
[799476555524210688].txt
[800965685333327892].txt [part 2].txt
[800965685333327892].txt [part 3].txt
[800965685333327892].txt [part 4].txt
[800965685333327892].txt
[803664746104225812].txt
[805557554730106910].txt
[807381864854716476].txt
[807382101501935636].txt
[807382113623998524].txt
[807382297103302717].txt
[807382307328491540].txt
[807735380677230645].txt
[812542570721705994].txt
[818310036131938324].txt
All files cleaned and saved in C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\C1
eaned Discord Data

```

Part 3 – AntConc Analysis of Discord Data and Urban Dictionary One-Word Slang List

Following Parts 1 & 2 above, the .txt files were converted to .db (database) files in AntConc.



AntConc - Corpus Manager

Corpus Database

Raw File(s)

Word List

Corpus Database Library

Add Database File(s)

Add Database Dir

Export Library

Import Library

View

Expand level 1

Show online corpora

Available Corpora (.db)	Status	Corpus Size
demo.db	Ready	1 MB
User (List)		
Cleaned_Discord_Data.db	Target	13016 MB
Contemporary Era.db	Ready	10961 MB
Digital Revolution.db	Ready	3755 MB
final_slang_words.db	Ready	52 KB
Golden Age Era.db	Ready	727 MB
New Hollywood Modern Era.db	Ready	1242 MB
Post War Era.db	Ready	901 MB
Silent Film Era.db	Ready	16 MB
Urban Dictionary Words.db	Reference	197 MB

☐ Connect online

Update

100%

Open As Reference Corpus

Status: The library is up to date.

Target Corpus

Reference Corpus

Corpus name

Dictionary Words.db

Swap (with target)

Description

Category	Description
full_name	Urban Dictionary Words
short_name	Urban Dictionary
file_count	1
token_count	1540197
type_count	1540197
encoding	utf_8_sig
token_definition	[p{L}]+
ignore_header	False
ignore_items	False
number_replace	False

Save (to file)

Close

Return to Main Window

AntConc - Global Settings

Category

Colors

Files

Fonts

Language Direction

Restore Settings

Searches

Statistics

Tags

Tool Filters

Tool Filters

Reset

☒ Cluster/N-Gram/Collocate/Word/Keyword/Wordcloud F

☐ Use words in file ☒ Hide words in file

Add File

Preview

Clear

English_stoplist.txt

Apply

The below image shows the resulting ‘keywordlist.csv’ file that was exported from AntConc.

The parameters for this list were minimum frequency of 200 and minimum range of 50.

	A	B	C	D	E	F	G	H	I	J
1	type	pos	headword	freq_tar	freq_ref	range_tar	range_ref	likelihood	effect	
2	i			11479231	0	736	0	168510.4	0.101	
3	a			5719020	0	736	0	82800.83	0.052	
4	you			3984891	0	736	0	57457.91	0.036	
5	n			3689422	0	736	0	53160.57	0.034	
6	the			3095485	0	736	0	44540.39	0.028	
7	s			3091457	0	735	0	44482.01	0.028	
8	o			2988740	0	735	0	42993.68	0.027	
9	to			2920586	0	736	0	42006.56	0.027	
10	it			2866390	0	736	0	41221.82	0.026	
11	t			2807932	0	736	0	40375.6	0.026	
12	e			2781181	0	734	0	39988.43	0.026	
13	is			2482169	0	736	0	35664.18	0.023	
14	u			2420121	0	736	0	34767.62	0.022	
15	m			2061384	0	736	0	29589.14	0.019	
16	and			1918460	0	736	0	27528.41	0.018	
17	that			1897913	0	736	0	27232.27	0.018	
18	my			1719037	0	736	0	24655.36	0.016	
19	h			1660648	0	734	0	23814.67	0.015	
20	me			1492351	0	735	0	21392.79	0.014	
21	in			1483086	0	736	0	21259.52	0.014	
22	what			1398308	0	735	0	20040.29	0.013	
23	r			1377264	0	735	0	19737.72	0.013	
24	l			1320801	0	735	0	18926.06	0.012	
25	d			1252797	0	736	0	17948.77	0.012	
26	y			1183200	0	735	0	16948.91	0.011	
27	can			1150393	0	736	0	16477.7	0.011	
28	like			1137753	0	735	0	16296.17	0.011	
29	for			1130255	0	735	0	16188.49	0.01	
30	of			1127832	0	736	0	16153.7	0.01	
31	im			1099153	0	736	0	15741.88	0.01	
32	no			1089017	0	736	0	15596.35	0.01	
33	have			1071522	0	736	0	15345.17	0.01	
34	are			1070773	0	735	0	15334.41	0.01	
35	don			1044309	0	736	0	14954.5	0.01	
36	do			1013548	0	736	0	14512.97	0.009	
37	on			970091	0	736	0	13889.3	0.009	
38	but			968414	0	736	0	13865.24	0.009	
39	not			965883	0	736	0	13828.92	0.009	
40	so			963981	0	736	0	13801.63	0.009	
41	just			933658	0	736	0	13366.54	0.009	
42	w			885471	0	735	0	12675.26	0.008	
43	how			882849	0	736	0	12637.65	0.008	
44	g			796198	0	735	0	11394.97	0.007	
45	if			785513	0	736	0	11241.77	0.007	
46	welcome			784070	0	729	0	11221.09	0.007	

Part 4 – WordNet Dictionary Processing



PRODUCTS SERVICES RESOURCES PRICING COMPANY BLOG

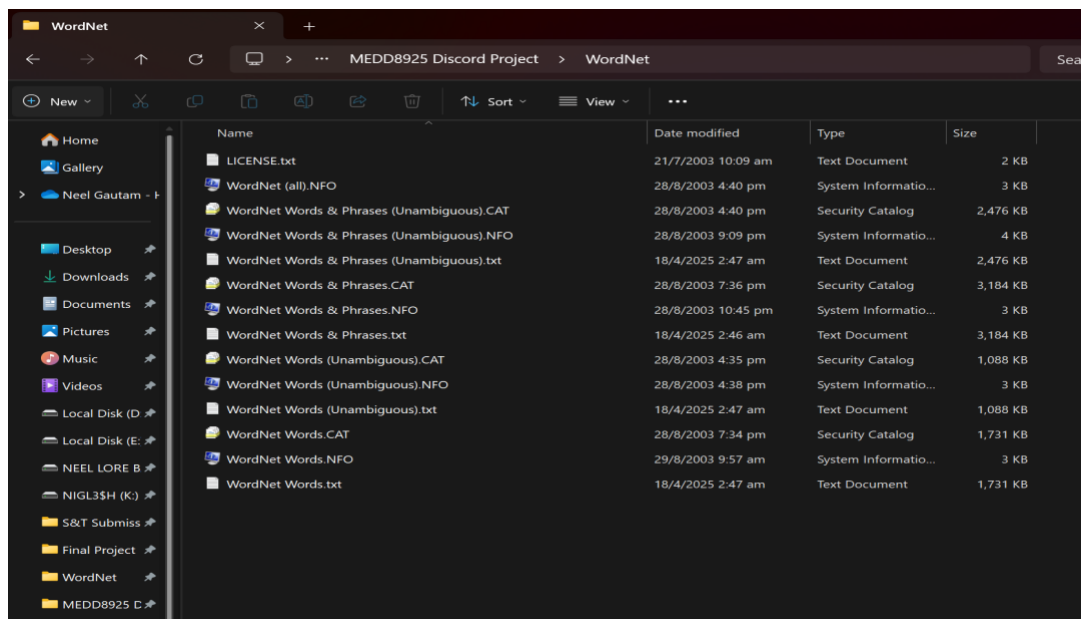
Book a web demo

Download Categorization Dictionary

Click [here](#) to download the four dictionaries (about 2Mb).

WordNet lexical categories and sample words					
DIMENSION	EXAMPLES	UNAMBIGUOUS WORDS	TOTAL WORDS	UNAMBIGUOUS WORDS & PHRASES	TOTAL WORDS & PHRASES
I. ADVERBS					
All	actually, deeply, fully	3233	3782	3782	4667
II. ADJECTIVES					
All		14273	17553		
Pertainyms	abyssal, genetic, intimal	2817	4209	2837	4247
Participial	kidnapped, pulled, sublimed	85	125	87	127

.CAT files were exported to .txt files in Notepad++



```
In [2]: import os
import re
import pandas as pd

# Directory containing the txt files
txt_dir = r"C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\WordNet"

# Find all relevant .txt files except LICENSE.txt
txt_files = [f for f in os.listdir(txt_dir) if f.lower().endswith('.txt') and "license" not in f.lower()]

entries = []

for filename in txt_files:
    with open(os.path.join(txt_dir, filename), encoding='utf-8') as f:
        cat = None
        subcat = None
        for line in f:
            line = line.strip()
            if not line:
                continue

            # Top-Level CATEGORY (all uppercase, no punctuation)
            if re.fullmatch(r'[A-Z ]+', line):
                cat = line
                subcat = None # Reset subcategory
                continue

            # SUBCATEGORY (all uppercase, possibly with dots, hyphens, or spaces)
            if re.fullmatch(r'[A-Z0-9\.\-\_ ]+', line) and not line.endswith(' '):
                subcat = line
                continue

            # WORD line (all uppercase, possibly with hyphens or underscores, ends with (n))
            m = re.match(r'^([A-Z0-9\-\_ ]+)\s+(\d+)\$', line)
            if m:
                word = m.group(1).strip()
                entries.append({
                    "word": word,
                    "category": cat,
                    "subcategory": subcat,
                    "source_file": filename
                })

# Build DataFrame
df = pd.DataFrame(entries)

# Remove duplicates
df = df.drop_duplicates(subset=["word", "category", "subcategory"])

# Export the main cleaned dataset
df.to_csv("all_wordnet_cleaned.csv", index=False)
print(f"Total entries (deduplicated): {len(df)}")

# Export a list of all unique categories and subcategories
categories = df[['category', 'subcategory']].drop_duplicates().sort_values(['category', 'subcategory'])
categories.to_csv("all_wordnet_categories.csv", index=False)
print("Categories exported to all_wordnet_categories.csv")

# Optional: Show a few rows as preview
print(df.head(25))
```

```
In [3]: import pandas as pd
from nltk.corpus import stopwords
import nltk

nltk.download('stopwords')

# --- File paths ---
wordnet_path = r"C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\all_wordnet_cleaned.csv"
keywords_path = r"C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\keywordlist.csv"
output_path = r"C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\classified_keywords.csv"

# --- Load Data ---
df_wordnet = pd.read_csv(wordnet_path)
df_keywords = pd.read_csv(keywords_path)

# --- Clean and Remove Stopwords ---
keyword_col = 'word' if 'word' in df_keywords.columns else df_keywords.columns[0]
df_keywords['lower_word'] = df_keywords[keyword_col].str.lower()
stop_words = set(stopwords.words('english'))
df_keywords = df_keywords[~df_keywords['lower_word'].isin(stop_words)]

# --- Prepare WordNet for Matching ---
df_wordnet['lower_word'] = df_wordnet['word'].str.lower()

# --- Merge/Join ---
merged = pd.merge(df_keywords, df_wordnet, on='lower_word', how='left', suffixes=('_keyword', '_wordnet'))

# --- Save Output ---
merged.to_csv(output_path, index=False)
print(f"Classified keyword list saved to: {output_path}")

# Preview some results
print(merged[['lower_word', 'category', 'subcategory']].head(20))
```

Below was the resulting output of the above code. Given it was messy with individual letters making their way in, more refinement was done on this ‘classified_keywords.csv’ file to prepare it for further analysis

AutoSave

Off

classified_keywords.csv

Saved to this PC

Search

FileHomeInsertPage LayoutFormulasDataReviewViewAutomateDeveloperHelpFoxit PDFAcrobat

Paste

Cut

Copy

Format Painter

Clipboard

Aptos Narrow

11

A⁺

A⁻

B

I

U

Font

Alignment

General

\$

%

Number

Conditional Formatting

Format as Table

Normal

Neutral

A1

type

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	type	pos	headword	freq_tar	freq_ref	range_tar	range_ref	likelihood	effect	lower_worc	word	category	subcategor	source_file		
2	n			3689422	0	736	0	53160.57	0.034	n	N	ADJECTIVE:ADJ.PERT		WordNet Words & Phrases.txt		
3	n			3689422	0	736	0	53160.57	0.034	n	N	ADJECTIVE:ADJ.ALL		WordNet Words & Phrases.txt		
4	n			3689422	0	736	0	53160.57	0.034	n	N	ADVERBS	ADV.ALL	WordNet Words & Phrases.txt		
5	n			3689422	0	736	0	53160.57	0.034	n	N	NOUNS	NOUN.TOP	WordNet Words & Phrases.txt		
6	n			3689422	0	736	0	53160.57	0.034	n	N	NOUNS	NOUN.ART	WordNet Words & Phrases.txt		
7	n			3689422	0	736	0	53160.57	0.034	n	N	NOUNS	NOUN.BOE	WordNet Words & Phrases.txt		
8	n			3689422	0	736	0	53160.57	0.034	n	N	NOUNS	NOUN.COM	WordNet Words & Phrases.txt		
9	n			3689422	0	736	0	53160.57	0.034	n	N	NOUNS	NOUN.OBJ	WordNet Words & Phrases.txt		
10	n			3689422	0	736	0	53160.57	0.034	n	N	NOUNS	NOUN.PER	WordNet Words & Phrases.txt		
11	n			3689422	0	736	0	53160.57	0.034	n	N	NOUNS	NOUN.PRC	WordNet Words & Phrases.txt		
12	n			3689422	0	736	0	53160.57	0.034	n	N	NOUNS	NOUN.QU/	WordNet Words & Phrases.txt		
13	n			3689422	0	736	0	53160.57	0.034	n	N	NOUNS	NOUN.REL	WordNet Words & Phrases.txt		
14	n			3689422	0	736	0	53160.57	0.034	n	N	NOUNS	NOUN.SUE	WordNet Words & Phrases.txt		
15	n			3689422	0	736	0	53160.57	0.034	n	N	NOUNS	NOUN.TIMI	WordNet Words & Phrases.txt		
16	e			2781181	0	734	0	39988.43	0.026	e	E	ADVERBS	ADV.ALL	WordNet Words & Phrases.txt		
17	e			2781181	0	734	0	39988.43	0.026	e	E	NOUNS	NOUN.COM	WordNet Words & Phrases.txt		
18	e			2781181	0	734	0	39988.43	0.026	e	E	NOUNS	NOUN.QU/	WordNet Words & Phrases.txt		
19	e			2781181	0	734	0	39988.43	0.026	e	E	NOUNS	NOUN.REL	WordNet Words & Phrases.txt		
20	e			2781181	0	734	0	39988.43	0.026	e	E	NOUNS	NOUN.SUE	WordNet Words & Phrases.txt		
21	e			2781181	0	734	0	39988.43	0.026	e	E	VERBS	VERB.BOD	WordNet Words & Phrases.txt		
22	e			2781181	0	734	0	39988.43	0.026	e	E	VERBS	VERB.CHA	WordNet Words & Phrases.txt		
23	e			2781181	0	734	0	39988.43	0.026	e	E	VERBS	VERB.COG	WordNet Words & Phrases.txt		
24	e			2781181	0	734	0	39988.43	0.026	e	E	VERBS	VERB.COM	WordNet Words & Phrases.txt		
25	e			2781181	0	734	0	39988.43	0.026	e	E	VERBS	VERB.COM	WordNet Words & Phrases.txt		
26	e			2781181	0	734	0	39988.43	0.026	e	E	VERBS	VERB.CON	WordNet Words & Phrases.txt		
27	e			2781181	0	734	0	39988.43	0.026	e	E	VERBS	VERB.CON	WordNet Words & Phrases.txt		
28	e			2781181	0	734	0	39988.43	0.026	e	E	VERBS	VERB.CRE	WordNet Words & Phrases.txt		
29	e			2781181	0	734	0	39988.43	0.026	e	E	VERBS	VERB.MOTI	WordNet Words & Phrases.txt		
30	e			2781181	0	734	0	39988.43	0.026	e	E	VERBS	VERB.PERC	WordNet Words & Phrases.txt		
31	e			2781181	0	734	0	39988.43	0.026	e	E	VERBS	VERB.POS	WordNet Words & Phrases.txt		
32	e			2781181	0	734	0	39988.43	0.026	e	E	VERBS	VERB.SOCI	WordNet Words & Phrases.txt		
33	e			2781181	0	734	0	39988.43	0.026	e	E	VERBS	VERB.STAT	WordNet Words & Phrases.txt		
34	u			2420121	0	736	0	34767.62	0.022	u	U	ADJECTIVE:ADJ.PERT		WordNet Words & Phrases.txt		
35	u			2420121	0	736	0	34767.62	0.022	u	U	ADJECTIVE:ADJ.ALL		WordNet Words & Phrases.txt		
36	u			2420121	0	736	0	34767.62	0.022	u	U	ADVERBS	ADV.ALL	WordNet Words & Phrases.txt		
37	u			2420121	0	736	0	34767.62	0.022	u	U	NOUNS	NOUN.COM	WordNet Words & Phrases.txt		
38	u			2420121	0	736	0	34767.62	0.022	u	U	NOUNS	NOUN.SUE	WordNet Words & Phrases.txt		
39	h			1660648	0	734	0	23814.67	0.015	h	H	NOUNS	NOUN.ATTI	WordNet Words & Phrases.txt		
40	h			1660648	0	734	0	23814.67	0.015	h	H	NOUNS	NOUN.COM	WordNet Words & Phrases.txt		
41	h			1660648	0	734	0	23814.67	0.015	h	H	NOUNS	NOUN.QU/	WordNet Words & Phrases.txt		
42	h			1660648	0	734	0	23814.67	0.015	h	H	NOUNS	NOUN.SUE	WordNet Words & Phrases.txt		
43	r			1377264	0	735	0	19737.72	0.013	r	R	ADJECTIVE:ADJ.PERT		WordNet Words & Phrases.txt		
44	r			1377264	0	735	0	19737.72	0.013	r	R	ADJECTIVE:ADJ.ALL		WordNet Words & Phrases.txt		
45	r			1377264	0	735	0	19737.72	0.013	r	R	ADVERBS	ADV.ALL	WordNet Words & Phrases.txt		
46	r			1377264	0	735	0	19737.72	0.013	r	R	NOUNS	NOUN.ATTI	WordNet Words & Phrases.txt		

Part 5 – Cleaning up of ‘classified_keywords.csv’

Irrelevant columns were dropped such as range_ref, lower_word, and source_file as they served zero value.

Jupyter Refined Classified Dictionary List Last Checkpoint: 15 hours ago (autosaved) Logout

```
In [3]: import pandas as pd

# File paths
input_path = r"C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\classified_keywords.csv"
output_path = r"C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\refined_classified_keyw

# Load CSV
df = pd.read_csv(input_path)

# Remove rows where 'type' is duplicated (keep first)
df = df.drop_duplicates(subset=['type'])

# Remove rows where 'type' is a single letter or blank/empty/NaN
df = df[df['type'].apply(lambda x: isinstance(x, str) and len(x.strip()) > 1)]

# Drop unwanted columns by name
columns_to_drop = [
    'range_ref',      # G
    'lower_word',     # J
    'lower_word',     # K (type in your sheet? adjust as needed)
    'source_file'     # P
]
df = df.drop(columns=[col for col in columns_to_drop if col in df.columns], errors='ignore')

# Save to new CSV
df.to_csv(output_path, index=False)
print(f"Refined file saved to {output_path}")

# Preview
print(df.head())
```

Refined file saved to C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\refined_classified_keywords.csv

	type	pos	headword	freq_tar	freq_ref	range_tar	likelihood \
50	like	NaN	NaN	1137753	0	735	16296.168
53	im	NaN	NaN	1099153	0	736	15741.881
61	welcome	NaN	NaN	784070	0	729	11221.085
110	get	NaN	NaN	624800	0	736	8938.409
121	oh	NaN	NaN	607688	0	734	8693.258

	effect	word	category	subcategory
50	0.011	LIKE	ADJECTIVES	ADJ.ALL
53	0.010	NaN	NaN	NaN
61	0.007	WELCOME	ADJECTIVES	ADJ.ALL
110	0.006	GET	VERBS	VERB.BODY
121	0.006	OH	NOUNS	NOUN.LOCATION

This resulted in the much cleaner ‘refined_classified_keywords.csv’ file below

AutoSave Off refined_classified_keywords.csv Saved to this PC

	A	B	C	D	E	F	G	H	I	J	K
1	type	freq_tar	range_tar	likelihood	effect	word	category	subcategory			
2	like	1137753	735	16296.17	0.011	LIKE	ADJECTIVE	ADJ.ALL			
3	im	1099153	736	15741.88	0.01						
4	welcome	784070	729	11221.09	0.007	WELCOME	ADJECTIVE	ADJ.ALL			
5	get	624800	736	8938.409	0.006	GET	VERBS	VERB.BODY			
6	oh	607688	734	8693.258	0.006	OH	NOUNS	NOUN.LOCATION			
7	good	604621	736	8649.321	0.006	GOOD	ADJECTIVE	ADJ.ALL			
8	yes	542445	734	7758.75	0.005	YES	NOUNS	NOUN.COMMUNICATION			
9	dont	517848	735	7406.509	0.005						
10	know	483496	735	6914.638	0.004	KNOW	VERBS	VERB.COGNITION			
11	one	469663	736	6716.592	0.004	ONE	ADJECTIVE	ADJ.ALL			
12	hug	460708	724	6588.391	0.004	HUG	NOUNS	NOUN.ACT			
13	hi	442385	732	6326.092	0.004	HI	NOUNS	NOUN.COMMUNICATION			
14	ok	405439	735	5797.268	0.004	OK	ADJECTIVE	ADJ.ALL			
15	go	405064	735	5791.901	0.004	GO	ADJECTIVE	ADJ.ALL			
16	got	404171	736	5779.12	0.004						

Part 6 – Further refinement of ‘refined_classified_keywords.csv’

 jupyter

Further Refined Keyword List of Slang

Last Checkpoint: 15 hours ago (autosaved)

 Logout

FileEditViewInsertCellKernelWidgetsHelp

Not TrustedPython 3 (ipykernel)

In [1]:

```
import pandas as pd
import re
```

In [2]:

```
# Replace with your actual file path
df = pd.read_csv('refined_classified_keywords.csv')
df.head()
```

Out[2]:

	type	freq_tar	range_tar	likelihood	effect	word	category	subcategory
0	like	1137753	735	16296.168	0.011	LIKE	ADJECTIVES	ADJ.ALL
1	im	1099153	736	15741.881	0.010	NaN	NaN	NaN
2	welcome	784070	729	11221.085	0.007	WELCOME	ADJECTIVES	ADJ.ALL
3	get	624800	736	8938.409	0.006	GET	VERBS	VERB.BODY
4	oh	607688	734	8693.258	0.006	OH	NOUNS	NOUN.LOCATION

In [3]:

```
# Keep if subcategory is 'NOUN.COMMUNICATION' or blank/NaN
mask = (df['subcategory'] == 'NOUN.COMMUNICATION') | (df['subcategory'].isnull()) | (df['subcategory'] == '')
filtered_df = df[mask].copy()
filtered_df.head()
```

Out[3]:

	type	freq_tar	range_tar	likelihood	effect	word	category	subcategory
1	im	1099153	736	15741.881	0.010	NaN	NaN	NaN
6	yes	542445	734	7758.750	0.005	YES	NOUNS	NOUN.COMMUNICATION
7	dont	517848	735	7406.509	0.005	NaN	NaN	NaN
11	hi	442385	732	6326.092	0.004	HI	NOUNS	NOUN.COMMUNICATION
14	got	404171	736	5779.120	0.004	NaN	NaN	NaN

In [4]:

```
def canonicalize_type(word):
    word = str(word).lower()
    # Remove Leading 'n' if followed by 'bruh' or 'good'
    word = re.sub(r'^n(bruh|good)$', r'\1', word)
    # Remove trailing 'n' if preceded by 'bruh' or 'good'
    word = re.sub(r'^(bruh|good)n$', r'\1', word)
    return word

filtered_df['canonical_type'] = filtered_df['type'].apply(canonicalize_type)
filtered_df.head(10)
```

Out[4]:

	type	freq_tar	range_tar	likelihood	effect	word	category	subcategory	canonical_type
1	im	1099153	736	15741.881	0.010	NaN	NaN	NaN	im
6	yes	542445	734	7758.750	0.005	YES	NOUNS	NOUN.COMMUNICATION	yes
7	dont	517848	735	7406.509	0.005	NaN	NaN	NaN	dont
11	hi	442385	732	6326.092	0.004	HI	NOUNS	NOUN.COMMUNICATION	hi
14	got	404171	736	5779.120	0.004	NaN	NaN	NaN	got
19	guys	319615	732	4569.182	0.003	NaN	NaN	NaN	guys
25	hey	293683	735	4198.208	0.003	NaN	NaN	NaN	hey
27	wanna	292759	734	4184.990	0.003	NaN	NaN	NaN	wanna
34	gonna	237783	733	3398.675	0.002	NaN	NaN	NaN	gonna
36	hello	235588	735	3367.284	0.002	HELLO	NOUNS	NOUN.COMMUNICATION	hello

In [5]:

```
# If your numeric columns differ, adjust the list below
agg_dict = {
    'freq_tar': 'sum',
    'range_tar': 'sum',
    'likelihood': 'sum',
    'effect': 'sum',
    'type': 'first',
    'word': 'first',
    'category': 'first',
    'subcategory': 'first'
    # Add any additional columns you wish to keep
}

# Group by canonical_type
result_df = filtered_df.groupby('canonical_type', as_index=False).agg(agg_dict)
result_df.head(10)
```

```
Out[5]:
```

	canonical_type	freq_tar	range_tar	likelihood	effect	type	word	category	subcategory
0	aa	3840	682	54.856	0.0	aa	AA	NOUNS	NOUN.COMMUNICATION
1	aaah	1598	494	22.828	0.0	aaah	None	None	None
2	aaahhh	2419	242	34.556	0.0	aaahhh	None	None	None
3	aboutn	4795	685	68.499	0.0	aboutn	None	None	None
4	abs	2981	566	42.585	0.0	abs	None	None	None
5	abt	21268	709	303.835	0.0	abt	None	None	None
6	abusing	2415	621	34.499	0.0	abusing	None	None	None
7	accent	7601	702	108.584	0.0	accent	ACCENT	NOUNS	NOUN.COMMUNICATION
8	accountn	2200	578	31.428	0.0	accountn	None	None	None
9	accounts	4888	695	69.827	0.0	accounts	None	None	None

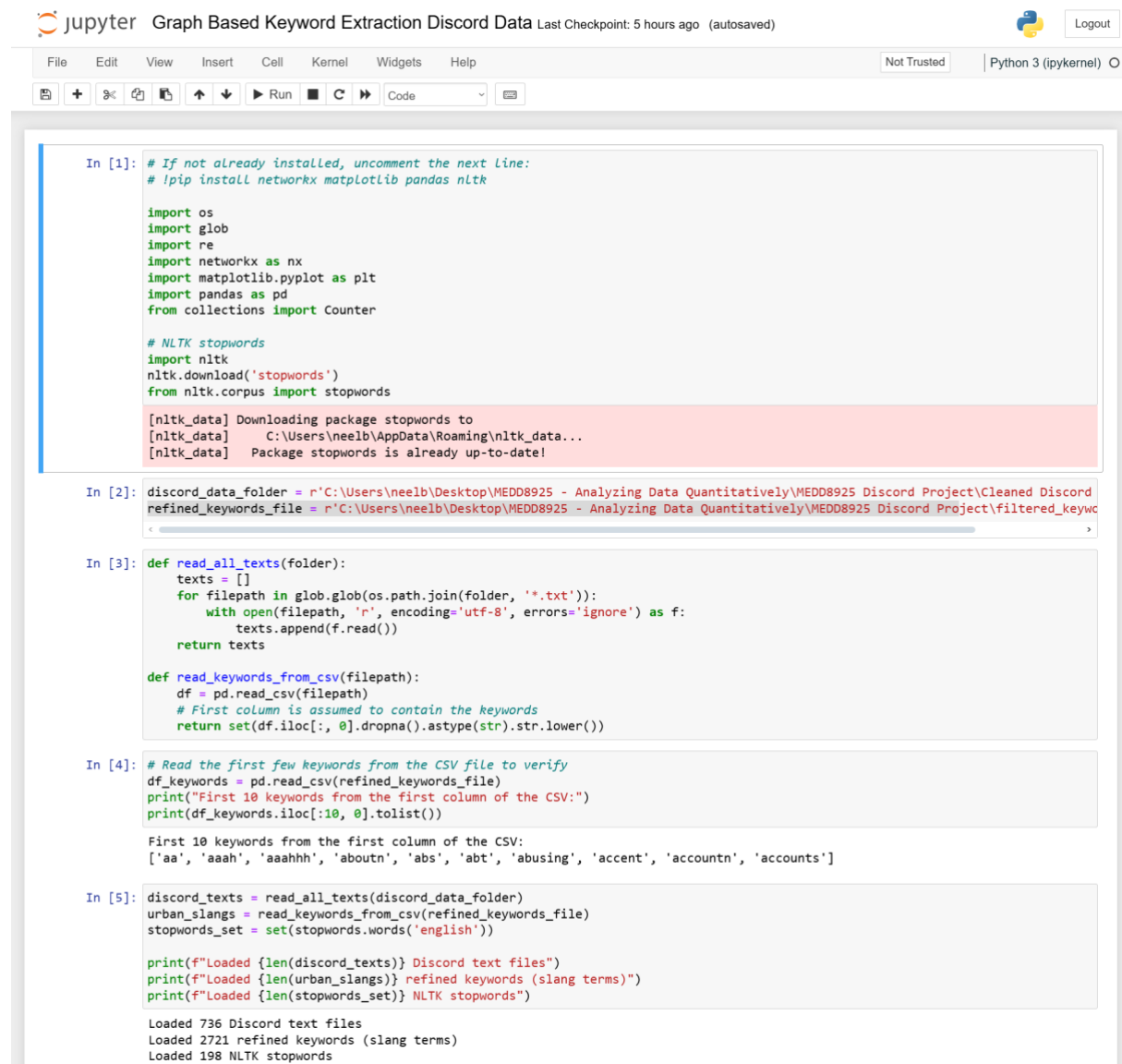
```
In [6]: result_df.to_csv('filtered_keywords_cleaned.csv', index=False)
```

This resulted in the 'filtered_keywords_cleaned.csv' file below which was far cleaner

	A	B	C	D	E	F	G	H	I	J	K
1	canonical_	freq_tar	range_tar	likelihood	effect	type	word	category	subcategory		
2	aa	3840	682	54.856	0	aa	AA	NOUNS	NOUN.COMMUNICATION		
3	aaah	1598	494	22.828	0	aaah					
4	aaahhh	2419	242	34.556	0	aaahhh					
5	aboutn	4795	685	68.499	0	aboutn					
6	abs	2981	566	42.585	0	abs					
7	abt	21268	709	303.835	0	abt					
8	abusing	2415	621	34.499	0	abusing					
9	accent	7601	702	108.584	0	accent	ACCENT	NOUNS	NOUN.COMMUNICATION		
10	accountn	2200	578	31.428	0	accountn					
11	accounts	4888	695	69.827	0	accounts					
12	ackerman	2725	195	38.928	0	ackerman					
13	actions	1962	590	28.028	0	actions					
14	activen	2441	576	34.871	0	activen					
15	acts	2120	631	30.285	0	acts	ACTS	NOUNS	NOUN.COMMUNICATION		
16	actuallyn	3947	675	56.385	0	actuallyn					
17	adan	1951	72	27.871	0	adan					
18	adding	3822	687	54.599	0	adding					
19	adios	1671	560	23.871	0	adios	ADIOS	NOUNS	NOUN.COMMUNICATION		
20	admin	24898	722	355.696	0	admin					
21	adminn	1917	515	27.385	0	adminn					
22	admins	5443	689	77.756	0	admins					
23	ads	5398	698	77.113	0	ads					
24	adults	2000	591	28.571	0	adults					
25	advice	7671	716	109.584	0	advice	ADVICE	NOUNS	NOUN.COMMUNICATION		
26	af	17666	724	252.374	0	af					
27	afk	46923	720	670.382	0	afk					
28	afkn	2292	496	32.742	0	afkn					
29	afn	4314	646	61.627	0	afn					
30	afternoon	6295	706	89.927	0	afternoon	AFTERNOO	NOUNS	NOUN.COMMUNICATION		
31	afton	1973	93	28.185	0	afton					
32	againn	18467	725	263.818	0	againn					
33	agen	2672	618	38.171	0	agen					
34	agent	1741	491	24.871	0	agent	AGENT	NOUNS	NOUN.COMMUNICATION		
35	ages	4882	700	69.742	0	ages					

Part 7 – Graph-based approach using PageRank

The below used the 2721 refined keywords and the NLTK stopwords library to create a ranked PageRank list and a co-occurrence graph of slang terms.



```
In [1]: # If not already installed, uncomment the next line:
# !pip install networkx matplotlib pandas nltk

import os
import glob
import re
import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd
from collections import Counter

# NLTK stopwords
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

[nltk_data] Downloading package stopwords to
[nltk_data]   C:\Users\neelb\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

In [2]: discord_data_folder = r'C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\Cleaned Discord
refined_keywords_file = r'C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\filtered_keyw

In [3]: def read_all_texts(folder):
    texts = []
    for filepath in glob.glob(os.path.join(folder, '*.txt')):
        with open(filepath, 'r', encoding='utf-8', errors='ignore') as f:
            texts.append(f.read())
    return texts

def read_keywords_from_csv(filepath):
    df = pd.read_csv(filepath)
    # First column is assumed to contain the keywords
    return set(df.iloc[:, 0].dropna().astype(str).str.lower())

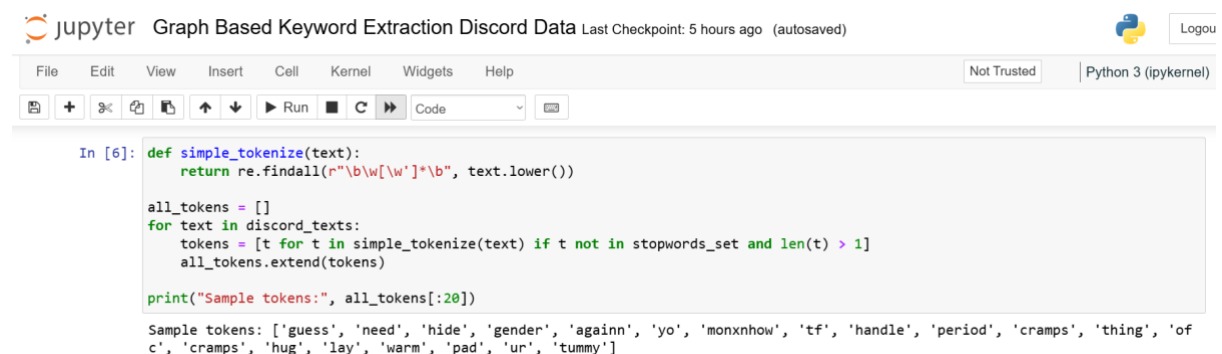
In [4]: # Read the first few keywords from the CSV file to verify
df_keywords = pd.read_csv(refined_keywords_file)
print("First 10 keywords from the first column of the CSV:")
print(df_keywords.iloc[:10, 0].tolist())

First 10 keywords from the first column of the CSV:
['aa', 'aaah', 'aaahhh', 'aboutn', 'abs', 'abt', 'abusing', 'accent', 'accountn', 'accounts']

In [5]: discord_texts = read_all_texts(discord_data_folder)
urban_slangs = read_keywords_from_csv(refined_keywords_file)
stopwords_set = set(stopwords.words('english'))

print(f"Loaded {len(discord_texts)} Discord text files")
print(f"Loaded {len(urban_slangs)} refined keywords (slang terms)")
print(f"Loaded {len(stopwords_set)} NLTK stopwords")

Loaded 736 Discord text files
Loaded 2721 refined keywords (slang terms)
Loaded 198 NLTK stopwords
```



```
In [6]: def simple_tokenize(text):
    return re.findall(r"\b\w+[w']*\b", text.lower())

all_tokens = []
for text in discord_texts:
    tokens = [t for t in simple_tokenize(text) if t not in stopwords_set and len(t) > 1]
    all_tokens.extend(tokens)

print("Sample tokens:", all_tokens[:20])

Sample tokens: ['guess', 'need', 'hide', 'gender', 'againn', 'yo', 'monxnhow', 'tf', 'handle', 'period', 'cramps', 'thing', 'of', 'c', 'cramps', 'hug', 'lay', 'warm', 'pad', 'ur', 'tummy']
```

```
In [7]: WINDOW_SIZE = 4
G = nx.Graph()

for i in range(len(all_tokens) - WINDOW_SIZE):
    window = all_tokens[i:i+WINDOW_SIZE]
    for j in range(len(window)):
        for k in range(j+1, len(window)):
            w1, w2 = window[j], window[k]
            if w1 != w2:
                if G.has_edge(w1, w2):
                    G[w1][w2]['weight'] += 1
                else:
                    G.add_edge(w1, w2, weight=1)

print(f"Graph has {G.number_of_nodes()} nodes and {G.number_of_edges()} edges.")
```

Graph has 3354222 nodes and 82280087 edges.

```
In [8]: pagerank_scores = nx.pagerank(G, weight='weight')
```

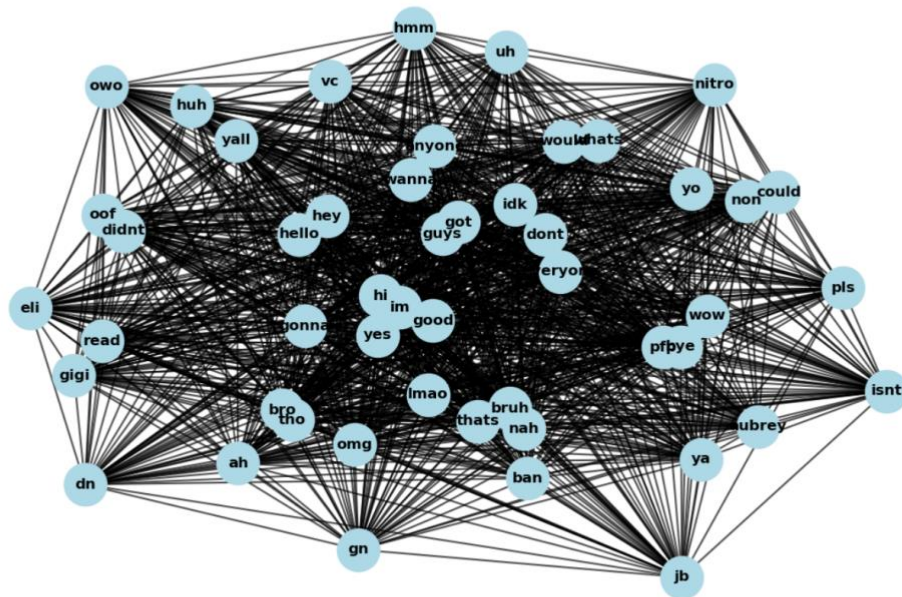
```
In [9]: central_keywords = [(word, score) for word, score in pagerank_scores.items() if word in urban_slangs]
central_keywords.sort(key=lambda x: x[1], reverse=True)

print("\nTop 25 central refined keywords used on Discord:\n")
for word, score in central_keywords[:25]:
    print(f"{word:15} {score:.5f}")
```

```
In [11]: top_n = 50
top_words = [w for w, _ in central_keywords[:top_n]]
subgraph = G.subgraph(top_words)

plt.figure(figsize=(12, 8))
pos = nx.spring_layout(subgraph, k=0.5, seed=42)
nx.draw_networkx_nodes(subgraph, pos, node_color='lightblue', node_size=1200)
nx.draw_networkx_edges(subgraph, pos, width=1.5, alpha=0.7)
nx.draw_networkx_labels(subgraph, pos, font_size=12, font_weight='bold')
plt.title("Network of Top Discord Refined Keywords")
plt.axis('off')
plt.tight_layout()
plt.show()
```

Network of Top Discord Refined Keywords



```
In [12]: # Convert to DataFrame and export to CSV
df_scores = pd.DataFrame(central_keywords, columns=['keyword', 'pagerank_score'])

# Save to CSV (choose your path/filename as needed)
output_path = r'C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\refined_keywords_pagerank_scores.csv'
df_scores.to_csv(output_path, index=False)

print(f"Exported PageRank scores for keywords to:\n{output_path}")
df_scores.head(10)
```

Exported PageRank scores for keywords to:
C:\Users\neelb\Desktop\MEDD8925 - Analyzing Data Quantitatively\MEDD8925 Discord Project\refined_keywords_pagerank_scores.csv

```
Out[12]:
```

	keyword	pagerank_score
0	im	0.008786
1	yes	0.004687