Sentiment Analysis with NLTK and Entity Linking

by

Jonathan Visona

B.S., Governors State University, 2000
B.A., Governors State University, 2009

GRADUATE CAPSTONE SEMINAR PROJECT

Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science

with a Major in Computer Science

Governors State University
University Park, IL 60484

2024

ABSTRACT

Sentiment analysis (SA) is an approach to natural language processing (NLP) for conducting semantic analysis in order to extract sentiment, opinion, intention, and ontology using various technologies exemplified by Python libraries such as Pandas and NLTK as well as natural language processing techniques including entity extraction, linking, and searching. This report provides a summary of exploratory research aimed at guiding future studies and developing a sentiment analysis research platform, categorizing it as R&D. The paper begins with conceptual analysis to situate the research in a particular metaphysical framework, that of NLP, artificial intelligence (AI), and artificial general intelligence (AGI), and then provides a motivation for the research, determines an appropriate scope, and explicitly formulates a research philosophy and methodology. In the research vein, after conducting a preliminary and cursory literature review of articles, it selects Bing Liu's SA framework in his book *Sentiment Analysis* and summarizes his most important conceptual foundations. In the development vein, the author explores the Kaggle Sentiment Analysis Dataset and solutions to a polarity classification problem which demonstrates a simple instance of an SA application using both formal NLP and ML strategies from a limited set of Python libraries. It also lays out a preliminary design and offers a GitHub repository with a prototype for a Python-based, formal systems object management framework as a potential tool for future research. Lastly, the author uses the findings of the exploratory research to analyze, synthesize, and speculate about SA as a basis for conducting research.

Table of Contents

1 Introduction to Sentiment Analysis Research

This research is designed to construct a foundation for acquiring an understanding of sentiment analysis (SA) in the broadest possible manner. While machine learning (ML) has become popular, it has limits compared to formal methods applied by a human to the task of semantic analysis, the central task of the subdiscipline of linguistic computation. This paper strives to understand how both formal and ML methods relate in the context of SA. Humans learn primarily through natural language, and ML strategies are a poor substitute for context-sensitive grammars. As Alan Turing, father of computer science states in "Computing Machine and Intelligence" one of the most important computer science articles ever written, "An important feature of a learning machine is that its teacher will often be very largely ignorant of quite what is going on inside, although he may still be able to some extent to predict his pupil's behaviour. This should apply most strongly to the later education of a machine arising from a child-machine of well-tried design (or programme). This is in clear contrast with normal procedure when using a machine to do computations..." The paper explores basic ML solutions (setting aside a detailed statistical exploration of foundation such as [1]) in conjunction with other tools for additional SA research that allows formal solutions also and lays out a multifaceted strategy for pursuing exploratory research as a basis of further doctoral-level investigation of SA within the metaphysical framework suggested by computer science researchers in [9] and [59].[1]

1.1 Exploratory Research

This exploratory research conducted about SA offers a preliminary map of a topic which has emerged as a discipline since about the turn of the century [28]. Roughly speaking, computer applications can be built to process text to determine some basic *non-logical* facts about the meaning of text, an aspect of what is called semantic analysis, including users' emotions, opinions, intentions, and ontological primitives.[2] The field of research that contains SA, originally called computational linguistics, is currently more often referred to as natural language processing (NLP), and falls under the discipline of artificial intelligence (AI) in computer science. The methods of AI are quite diverse.[3] Today, in recognition that AI technology has failed to achieve Turing's dream of building intelligence of the same capacity as human intelligence or the inability for anyone to design and implement a system that passes Turing's imitation game, AI is now held as distinct from artificial general intelligence (AGI) which constitutes a more philosophical and speculative endeavor still rejected by many thinkers though there are now scores of AGI research institutes all over the world complementing the traditional AI research conducted.[4]

The original AI researchers in the 1950's and 1960's at MIT, Stanford, and other universities were excessively optimistic about realizing Alan Turing's dream of building learning machines. One of the early papers with such optimistic authors that put forth a thesis that physical symbols systems were sufficient to realize such intelligence is the physical symbol system hypothesis (PSSH) of Newell and Simon.[5] After a number of claims failed to materialize regarding quick progress to replicating human

[1] These two seminal works, undertaken by two distinct doctors of CS, the philosophy of computer science is a relatively nascent interdisciplinary community.

[2] 'Ontological primitives' is a term that comes from ontology, one of the major disciplines in philosophy that revolves around answering questions like 'What is a thing?' and 'What things are there?' Natural language is full of ontological primitives and collectively they are known as a natural language ontology, which is distinct from the applied ontologies AI researchers build. The latter is a formalization of the former. In computer science, data-centric terminology often uses the terms 'entities', 'attributes', and 'relations'. Philosophers often use the term 'object' for 'entities' and 'properties' for 'attributes [35].

[3] For an encyclopedic look into AI, it is difficult to outdo [44].

[4] Many researchers also use the terms 'narrow AI' and 'broad AI' for AI and AGI, respectively. See the National Science Foundation's list of current AI research institutes at [https://nsf-gov-resources.nsf.gov/2023-08/AI_Research_Institutes_Map_2023_0.pdf] for a peek at how pervasive AI and AGI research has become in the US. A number of vocal advocates of AGI have published research in a compendium [17].

[5] Their article "Computer Science as Empirical Enquiry: Symbols and Search" represents a position that emphasizes the role of symbolic grammars in AI [37], and has been defended by John Haugeland who refers to the symbolic approach as GOFAI: good, old-fashion AI [20]. Connectionism is the other approach and

intelligence, the philosopher Hubert J. Dreyfus while working for RAND wrote a paper and then followed up with a book expressing skepticism and highlighting the deficiencies of AI applications [12].[6] It should be noted, however that machine learning (ML) algorithms have helped to close the gap between dream and reality, with IBM building both Deep Blue and Watson, both of which dethroned the world's greatest champions in chess and Jeopardy!, respectively; once again, famous and accomplished technical experts have begun talking about machines becoming intelligent like people, or even exceeding human intelligence and control, Ray Kurzweil being a famous proponent of such ideas.

Despite the strengths of big data, data mining, and ML in advancing AI technologies, they have not significantly closed the gap. [27] highlights the ongoing myths surrounding AI and AGI. Neither big data nor ML, as transformative as the techniques have been over the last 75 years, have been the magic bullet to turn the dreaming of the early AI pioneers into physical reality. The hype over AI, however, still has not died down. Embarrassingly, even seasoned computer engineers and researchers sometimes make hyperbolic claims.[7] It is therefore an overarching intent of this paper to keep in mind what specific formal symbolic and connectionist approaches are available for SA not just as an inventory for inventory's sake, but to use as a semantic toolbox to accept or reject claims about what AI and AGI can and cannot do; SA which on its face might be interpreted as showing aspects of human intelligence must be understood as doing so only in a manner that falls far short of replicating human cognition. Therefore, the author advances the claim that SA is a very useful set of methods in NLP used in industry and academia for exploring what people think, believe, and feel, and helpful in inferring what people will do, and SA, like the recent successes of LLMs, has not set us on a path of realizing human-level intelligence in digital systems in the foreseeable future. The author also briefly examines what SA contributes to the broader ambitions of AGI theorists, and a rationale for rejecting AGI in the spirit of Dreyfus and Larson, is given.

1.1.1 AI and NLP

As previously mentioned, the strategies available to an AI researcher are almost uncountable. This research addresses just the aspect of AI called NLP with a further reduction of scope since SA is only one facet of NLP. Note also, NLP is an interdisciplinary field since it relies heavily on mathematical logic, linguistics, philosophy of linguistics, and philosophy of language. For instance, only recently are comprehensive pictures of the full range of linguistic phenomena and their interconnections being put forward to inspire future NLP research.[8] In fact, a reading of philosophy of linguistics [47] and philosophy of language [52] reveals just how many open issues there are in understanding the essential mechanisms of language and semantics. As pointed out by Liu, the linguistic tradition often falls short of operationalization [28], and therefore, practical systems are the domain of the computer scientist and software engineer.

The SA researcher must also remember that besides a familiarity with natural language, two additional classes of algorithms pervade the practical implementation of SA. The first are the formal systems and semantics of those that rely on first-order logical languages or grammars like those developed by

relies on computational strategies and is based on statistical methods and artificial neural networks and rose to a particular prominence in the late 1980's with the two-volume publication known as *Parallel Distributed Processing* (PDP). Today, many researchers acknowledge the importance of considering both formal symbolic and statistical and probabilistic techniques as in [33] to effect practical solutions.

[6] Dreyfus claims his colleagues shunned him after his questioning of AI research. A more contemporary skepticism of bold AI claims can be found with [27]. He provides a thorough argument of why big data is insufficient to realize AGI, and the argument revolves around an understanding of the distinctions among deductive, inductive, and abductive reasoning.

[7] See Interesting Engineering's article "Fired engineer who called Google AI 'sentient,' warns Microsoft Bing a 'train wreck'" for an example of the hyperbole and misunderstanding of AI at [https://interestingengineering.com/innovation/engineer-google-warns-bing-ai-train-wreck].

[8] One model that takes a middle road between Chomsky's minimalist program and more speculative approaches like cognitive linguistics is that of Ray Jackendoff in [23]. He examines language starting at the phonological level and then moves upwards through syntax, semantics, and then concepts, pragmatics, and metaphysical relations. While much of SA is currently conducted at the level of the lexeme up to passages, in theory, lower-level analysis like morphemic analysis, phonemes, and even non-grammatical symbols could be used to supplement current approaches.

Richard Montague[9]; the second, often referred to as statistical or probabilistic, are those algorithms and heuristics that will be called computational intelligence.[10] Thus, SA is not a simple, single collection of algorithms, but an extremely broad programme and methodology for creating information from data, and then knowledge from information, and essentially encompasses many domains of NLP. In such a manner, SA is not a subset of NLP, but an aspect of NLP applied to a particular goal: understanding sentiment, opinion, intention, ontology, etc.[11]

Therefore, SA succeeds or fails based on the sophistication of the NLP and the range of formal and computational methods used by a practical, hybrid system. That means a thorough knowledge of syntactic phenomena of English as in [32] is just as important as being familiar with the formalisms that underlie various NLP ML strategies as in [18]. There are no hard borders among the domains of computer science, mathematical logic, linguistics, and philosophy, and the more interdisciplinary the approach to implement the SA solution, the more sophisticated the SA.[12]

1.1.2 Type Theory, NLP, and SA

[14] thoroughly introduces NLP and builds up to brief coverage of SA in Chapter 4, Section 4.1 Sentiment and Opinion Analysis. Concepts and techniques used in NLP are the core of SA. It is not strictly necessary, but a survey of statistical methods helps to make NLP more comprehensible. NLP is qualitatively different from other computer science approaches to computation which tend towards highly mathematical, logical, and set-theoretic natures whose semantic analysis is often grounded in operational, denotational, and axiomatic approaches semantics as briefly described by [40]. As an aside, Frege's program of logicism and subsequent Tarskian semantics, the received view among some researchers on semantics, relies heavily on ZFC today among mathematical logicians and falls short in providing an adequate basis for modeling natural language which is why the work of Richard Montague is important.[13]

Another extension of using ZFC, PA, proof theory, and model theory is recent developments in type theory and type-theoretic semantics.[14] For the undergraduate in computer science, [16] is a must-read for the basics of extending lambda calculus into a broader type theory. [40] provides a rigorous and fascinating introduction into type theory (and forms of mathematical induction and recursion incidentally). The author develops a foundation for understanding compiler typing systems, and while useful for understanding programming languages and compilers, also serves as a basis for understanding NLP, since there is an obvious conceptual relationship between types and classes. It has long been a practice in certain formal circles following improvements on type theory by L.E.J. Brower's students, to fortify type theory with an "intuitional framework" to move from intuitive definitions of collections to rigorous comprehensions of types in a theory called intuitionist type theory (ITT) which was invented by Per Martin-Löf [13].

While ITT is an advancement into intuitionism and incidentally a plausible psychologistic interpretation of linguistic formal semantics including mathematics, it does not offer many formalisms to describe

[9] Linguistic formal semantics is an approached pioneered by Richard Montague and advocated by Barbara Partee. Since the 1970's, a small, intrepid group of academics have been trying to bring together the best of both mathematical logical and linguistic formalisms. A good introduction can be found in [7].

[10] Per [26] various approaches to algorithms can be separated into non-formal techniques such as artificial neural networks, genetic algorithms, swarm intelligence, machine learning, fuzzy systems, and other statistical and probabilistic approaches. Each domain is a separate field of graduate-level expertise giving the SA researcher a cornucopia of strategies.

[11] Carnap called such language in normal discourse the 'formal mode'. Quine in his *Word & Object* preferred to call discourse to negotiate the particulars listed above as 'semantic ascent'. Given the back and forth meant to resolve ambiguities between two distinct speakers, it's fair to hypothesize that SA systems will only be optimized by if they include erotetic logics, which are formal methods focusing on the logic of questioning and answering.

[12] … and therefore, the more sophisticated the NLP and therefore the more sophisticated the AI.

[13] The author asks the philosophically inclined reader to accept the premise provisionally as logicism and neologicism are still popular among various tribes of mathematically inclined researchers [55] when it comes to language.

[14] [8] provide an overview of foundational issues in type-theoretical semantics, a discipline that uses type theory to metaphysical ground natural language semantics.

natural language. Dr. Arne Ranta, among others, achieved this by the application of formal semantics of mathematical logic, linguistics, and computer science in [42] with his type-theoretical grammar. For the astute student of linguistics, this is a framework that can facilitate projects related to dynamic semantics [39]. Briefly, linguistic formal semantics blends the best of formal logical methods with broader linguistic modeling from linguistic research, and Ranta provides not only a theoretical basis, but a practical tool set for simulating context-sensitive grammars with Grammatical Framework [43].[15]

1.1.3 SA, Experimental Philosophy, and the Rejection of AGI Claims

Since the advent of the electromechanical computer, the earliest computer scientists envisioned a future in which machines could learn from people and the world around them. For some, the interest in learning machines revolves around processing perceptual data be it auditory or visual. But for others, the pursuit is processing natural language with the goal of building up the equivalent of a theory of mind. This exploratory research about SA obviously approaches ML as the latter. Many facets of computer science hold allure to the researcher, but the holy grail of computer technology for many AI researchers, is using machines to replicate the intelligence of people, a field named “artificial intelligence” by pioneer John McCarthy back in the Dartmouth conference in the summer of 1956.[16]

Attempts to draw parallels between computers and the brain were not exclusive to Turing. Another computer science pioneer, John von Neuman conducted his own comparison in [61]. In fact, the chief defining characteristic of AI has always been to model human cognition; in fact, cognitive psychology’s displacement of behaviorism was partially driven by the rationalism of Chomsky and his approach to grammars and the evolution computer architecture, von Neumann having played a role by documenting the work of Mauchly and Eckert in the R&D of ENIAC early on [31].

Another notable contribution to the field of AI research comes from [46] with their attempts to build an early version of a system of AI agents. These sorts of projects still enliven industry with many organizations attempting to evolve simple regex chatbots into full agents of conversational AI using NLP methods. While more marketing than research, consider the call to action for building conversational AI in [63]. Proponents of hyperbolic AGI claims often latch onto these aspirations and make astonishing assumptions as [27] lays out. It therefore falls onto the current NLP research to explain why these claims should be rejected.

Since Brentano’s introduction [5], intentionality and judgment have evolved the philosophy of mind very thoroughly.[17] Today, philosophy and computer science have a healthy relationship in the form of experimental philosophy [25] with the empirical methods of computer science helping to strengthen the rational methods of philosophy. One example of this is how computer scientists build applied ontologies to understand how natural language ontology functions. One can consider computer science engaged in a fundamental programme of research with philosophy where the mathematical logical techniques and broader array of linguistic formal semantics attempt to model and build productive architectures of reason in the spirit of [2].

NLP has and must continue to look to linguistics and philosophy of mind for insights into methods. To that end, SA should be more than a collection of ad hoc industry projects, and should maintain a reciprocal relationship with the broader research communities involved in cognitive science

[15] Both computational and formal methods rest on notions of mathematical logic and computability where [4] should be required for all computer science graduate students.

[16] For a historical peek provided by Nils Nilson in the organization, purpose, and activities of the conference held in the summer of 1956 at Dartmouth College in Hanover, New Hampshire, read section 3.2 [38].

[17] For computer scientists unfamiliar with the philosophical foundations of AI, consider reading [21].

research.[18] Philosophers like Searle, Dennett, and Shea should be read to help to shape future NLP technologies. For instance, in [48] a basic look at expanding linguistic explicatures to implicatures and other forms of pragmatics is useful in understanding non-syntactic processing. Since later Wittgenstein's *Investigations* [65], the notion of a language-game has influenced countless thinkers about natural language including other influential philosophers like Wilfred Sellars and cognitive scientists working on broader theories of communication such as Sperber and Wilson with their influential relevance theory [51].[19]

Therefore, the astute computer scientist can safely reject hyperbolic AGI claims since it is clear that the formalist symbolism and the techniques of computational intelligence combined fall short of meeting the conditions of sufficiency and necessity for achieving adequate models of human-level intentionality as per [49] and [11]. Clearly the context-sensitive complexity of natural language has not been modeled, even with the addition of LLMs, and ML does not rise to the complexity and spirit of human learning [22]. The collective techniques and models of computer science fall short in a systemic way of achieving the sort of cognitive organization required to maintain mental or cognitive representation such as theorized by [10] and [50].

1.2 Problem and Motivation of SA Research

Having situated the topic of SA in NLP and AI, it now is necessary to formulate the problem this research intends to address:

> **Research Problem: Explore the nature of SA and related strategies in non-logical aspects of semantic analysis by 1) understanding its relationship to NLP, AI, and computer science more broadly, 2) describing a methodological philosophy and framework for accumulating SA knowledge, 3) selecting and distilling important features of an SA theoretical framework, 4) acquiring practical knowledge of a simple SA problem, 5) developing a simple technical platform for future SA studies, and lastly 6) reviewing, analyzing, synthesizing, and speculating about the SA research performed and future strategies for the continuation of SA research at the graduate level.**

If these 6 criteria are met with a quality effort, then the exploratory research should be viewed as successful. The author will briefly assess the results in Section 4: Research Analysis, Synthesis, and Speculation. The reader must judge for themselves whether this paper is a persuasive account.

SA represents a realistic and practical extension of AI research that speaks to a very limited, and still not fully charted territory of academic research and industry development, and succeeds to the extent it uses natural language to predict some important facts about the state of mind of the user by their language. By looking at sentiment, opinion, intention, and ontology, SA provides a window into the experiential state of mind of a person and allows AI to anticipate and act for the user, be they a customer or polling respondent, practically. While SA and the simplistic theory of mind it constructs doesn't rise to the level of a shrewd psychologist (or even a dog for that matter), it does allow computer scientists to provide yet another tool to automate some aspect of what books, dogs, and mere calculators cannot do; in this sense, it is an important tool in researching AI and the philosophy of mind.

[18] A good introduction comes from our British cousins in [6]. Cognitive science, for the unaware, is an interdisciplinary effort to use statistical, formal, and scientific methods to make sense of human cognition. While philosophically controversial for some, the mind should be understood as a product of the brain and the wider body. This latter thesis is common in the physicalism of scientific realism and the position of embodied cognition as in [60].

[19] Relevance theory, born of the research of cognitive science explains how explicatures of language are married with mechanisms of pragmatics, such as ostensive-inferential contextualization.

A number of explanatory unknowns are detected by SA research, and a common mantra found throughout [28] is "more research in this topic" needs to be done.

Besides the practical merits of having systems analyze text for such high-level aspects of human experience such as predicting markets, gathering data about democratic opinion, finding and fixing sources of bad products, and anticipating users' needs, developing successful SA validates hypotheses about human language and cognition and operationalizes linguistic theories; where systems work, we can have increased confidence that our scientific understanding of language and cognition is accurate and the fewer explanatory gaps confront us in scientific explanation and engineering solutions. Therefore, SA research is an important middle ground between mere statistical methods of data science for practical ends and lofty ambitions to build AGI systems that approach human-level intelligence (whatever that might mean).

1.2.1 Determination of Scope

One of the challenges of exploratory research is determining where to circumscribe the research. As an undergraduate, one is handed a textbook and told by way of syllabus what should be learned, and quite often prescriptions of how the material should be learned accompany the "roadmap" of knowledge required in the form of assignments and exercises. In exploratory research, there are no homework assignments and there is no syllabus. Thus, to limit the scope of this project, four motivations have been selected, two of them proximal to the practical, technical aspects of SA, and two of them build a better awareness of how SA relates within the broader framework of NLP, AI, and philosophy. In this way, balance between theory and practice is maintained.

1.2.2 Proximal Motivations

Two proximal motivations exist to narrow the scope of the inquiry:

1. **Explore the nature of a specific SA solution and related strategies.**
2. **Build a platform for long-term, on-going research into SA, intention mining, entity extraction and linking, and other NLP.**

The first proximal motivation is to probe the nature and boundaries of SA and understand its structure going beyond the Fregean project of reducing language to logical systems by finding techniques situated in non-formal methods.[20] Clearly not all semantic analysis need be formal. Since SA is a broad field that touches on all topics NLP touches on, the author builds out a basic vision of epistemological progression to reach a level of competence to enable more research in SA. In other words, the first proximal motivation envisions what a self-imposed curriculum to move in the direction of doctoral thesis would entail using practical tools on practical problems. To meet this goal, the author selected the Kaggle dataset for Sentiment Analysis.

The second proximal motivation serves the first. Mastery of a conceptual topic often revolves around a personal management system devoted to organizing the information for acquisition, retrieval, analysis, and synthesis of relevant information and resources. In the ideal, a research can use an electronic Zettlekasten[21] schema for organization; such a system that allowed for rapid implementation and testing of source code would be an aid. In this way, building a Python-based system to evaluate prototypes expedites research and opens up avenues of collaboration.

[20] Gottlob Frege is the father of analytical philosophy and the first modern proponent of reducing natural language to logical formal systems in order to provide a logical basis for mathematics [54]. This programme, known as logicism, was part of the anti-psychological drive to show the objectivity of the formal science of logic and was added to and improved on by many logicians and mathematicians. The logician Richard Montague in the 1970's added a deeper dimension with what has become known as Montague Grammar [24]. Again, formal methods rely on symbolic manipulation as opposed to connectionist, statistical, or probabilistic methods.

[21] Zettlekasten is an organizational strategy every academic should be familiar with. See [https://en.wikipedia.org/wiki/Zettelkasten].

1.2.3 Distal Motivations

There are also two distal motivations exist to narrow the scope of the inquiry:

1. **Explore how SA fits in the greater scheme of NLP strategies.**
2. **Understand how both formal techniques as well as computational strategies inform NLP research.**

The first distal motivation is about establishing the relationship between SA and NLP to better understand which NLP methods are not relevant to SA. While SA is intimately a part of NLP, many topics and techniques in [14] may go beyond the scope of SA. Only through a mutual exploration of SA and NLP simultaneously can a comparison and contrast be successful.

The second distal motivation is to understand how any techniques used to accomplish SA are situated in NLP more generally. For example, linear classification is a method that can be used across a variety of NLP tasks that are not part of SA. Discovering how such strategies can exist independently of SA helps to build a better picture of what SA is and is not.

1.3 Research Methodology

> **One of the most important distinction between two types of laws in science is the distinction between what may be called… empirical laws and theoretical laws. Empirical laws are laws that can be confirmed directly by empirical observations.**
>
> **Rudolf Carnap,** ***An Introduction to the Philosophy of Science***

> **Through deliberate and conscious investigations and experimentations, discovery and interpretation of evidence have constantly probed human existence, generating theories or laws in the bid to establish new sets of ideas, or real-world application of such new ideas, and/or revision of existing theories or principles. This systematic inquiry that aims to discover new things and create new facts or knowledge spans all facets of life.**
>
> **Mbanaso et al.,** ***Research Techniques for Computer Science, Information Systems and Cybersecurity***

Section 1.1 has established the metaphysical paradigm, and a problem statement has been formulated in Section 1.2, so it is now necessary to make explicit the nature of the details of the research as per Chapter 6: Research Philosophy, Design, and Methodology of [30]. This research provides an overview of types of techniques used with a brief description of the specific activities that satisfy each method in terms peculiar to this project.

1.3.1 Overview of Methods

In order to advance knowledge to serve as a foundation for seeking out explanatory gaps and filling them in future research, this paper has been built around a distinct set of research methods that explore both theory and practice superficially since this research is exploratory. The following are those methods:

1. **Framework Selection**
   a. **Philosophical and Linguistic Theory: Choosing appropriate theories and language communities.**
   b. **Technical Strategy: What sort of technology-practice is viable.**
2. **Comparative Studies**
   a. **Literature Review: Results from multiple studies to identify problems, themes, and resources.**

   - b. **Solutions Analysis: Combining results from multiple studies to identify patterns or overall effects.**
3. **Empirical Methods**
   - a. **ML Case Study: In-depth analysis of a single ML solution to explore its complexities.**
   - b. **Python Object Management Case Study: In-depth analysis of a single python framework solution to explore its complexities.**
4. **Design and Development**
   - a. **Prototyping: Building early versions of systems to test concepts and gather feedback.**
   - b. **Software Engineering: Applying systematic approaches to the design, development, and maintenance of software.**

1.3.1.1 Framework Selection

Research is conducted and communicated in language. The rules of language, however, are subject to the watchful eyes of the research community and the language they accept. Particularly important in research is adjudication of terminology, since terminology helps to define the ontological structure of the dominant research paradigm.[22] Therefore, by adopting the framework of an established researcher like Bing Liu, it quickly aides exploratory research by establishing an acceptable lexicon and research ontology. By adopting the framework in the form of an evidence-based book authored with the intent of summarizing the relevant literature, this author gains a reliable list of references.

Besides the theoretical framework, it also behooves the exploration to establish which technical framework can be used. As Python is the most popular programming language[23], and it is considered an important tool for many researchers because of its libraries. See [34] for an introduction to the utility of Pandas, NumPy, and Ipython as well as the Dframe class. [56] gives a brief, by highly effective and accessible introduction to ML theory and practice, and finds an opposite of sorts in [18] which is highly mathematical and abstract outside of the brief introductory passages proceeding each type of ML architecture. The best foray into ML was conducted by reading substantial portions of [36] which does an excellent job of balancing out theory and guidance on implementing Python-based solutions, including an introductory bag-of-words SA example. Given Python's popularity, the Kaggle Sentiment Analysis Dataset was a natural progression to a real-world example.

1.3.1.2 Comparative Studies

First, a preliminary literature review of 5 papers which rank at the top of a Google Scholar search with the string 'sentiment analysis' has been conducted. (See Section 1.4.1: Cursory Literature Review). This oriented the exploration, and helped establish Bing Liu's work *Sentiment Analysis* [28] as the book-length framework for this exploration. Both works helped to define common language and themes prevalent in SA.

Both the reviewed papers and the book helped to establish themes, and define the problems of SA. While the papers vary in their approaches and topics of analysis, they set the tone above and beyond tertiary and secondary sources on SA. They help to establish the general nature of the problem, and a range of solutions and concerns surround the topic. Having provided inroads into

[22] Here, Thomas Kuhn in his *The Structure of Scientific Revolutions* will be taken uncritically, and the terminology paradigmatic and normal science will be embraced. It is necessary for the individual to align themselves to normal science and establish credibility before any attempts to influence the paradigm are possible.

[23] See [https://www.tiobe.com/tiobe-index/]. At the current time, Python has a rating 22.85% is far more popular than both C++ and Java together whose combined rating is only 20.24%.

the theory, the viability of using Python and its libraries was established with the help of examining a range of solutions offered for the Kaggle Sentiment Analysis Dataset challenge. See Section 2 for more information on how particular Python libraries were selected.

1.3.1.3 Empirical Methods

The Kaggle Sentiment Analysis Dataset which is one of the many datasets available on a website devoted to data science and ML provided not only data derived from Twitter messages, but also a range of solutions offered by various data scientists and ML engineers. By examining this real-world example, it became possible to see a selection of rudimentary solutions to an SA problem which helped guide the construction of the author's solution. Such a solution provided practical insights, and the source code can be found in Appendix A.

A second attempt at creating an empirical dimension to this exploration was building a Python-based object management framework for future SA projects. The OO design implemented some central logic for creating and managing Python objects of arbitrary classes as well as a tkinter-based GUI with CLI integrated into the interface. The ambition was to transition from Jupyter notebooks to SA scripts extracted from the IPYNB format with the eventual goal of being able to programmatically automate model construction, evaluation, cross-validation, and parameter optimization. One semester proved insufficient for full completion of various goals, including metaprogramming functionality and an interface with a DBMS.

1.3.1.4 Design and Developing

Having selected the Kaggle Sentiment Analysis Dataset challenge and Python, VS Code and appropriate extensions were chosen and imported to allow for the use of Jupyter notebooks. A virtual environment was created and the software used in Section 1.3.1.3 was downloaded or written as well as executed. These prototypes helped to guide analysis and provided a real-world sample for grounding comprehension of principles.

Additionally, a GitHub repo was created, and git was used to track development, particularly of the object management framework named Montague. This practice furthers the goal of creating a GUI and a library of objects for rapid application development, and included features like automatically generating several types of objects related to SA such as executable scripts, blocks of code, and various textual data. Development was curtailed by time constraints. An equal amount of time was devoted to a custom solution to the Kaggle challenge and the development of Montague.

1.4 SA Conceptual Analysis

> **Emotions are described as turbulences in the stream of consciousness, the owner of which cannot help directly registering them; to external witnesses they are, in consequence, necessarily occult. The are occurrences which take place not in the public, physical world but in your or my secret, mental world.**
>
> **Gilbert Ryle, *The Concept of Mind***

> **[S]entiment is defined as an attitude, thought, or judgment prompted by feeling, whereas opinion is defined as a view, judgment, or appraisal formed in the mind about a particular matter.**
>
> **Bing Liu, *Sentiment Analysis***

When entering new research territory, it helps to establish a preliminary conceptual map for understanding the topic. The author progresses through a cursory literature review of articles, followed by adopting an established framework from an expert in the field. The former practice helps determine the latter. Since the author harbors aspirations for doctoral research, he chose the framework in question not merely on the merits of the credentials of the expert and the popularity of his book, but also to enable future emulation and collaboration with the expert who resides at a university near to the author.

1.4.1 Cursory Literature Review

In order to orient oneself in a new discipline, it is necessary to conduct literature review. Here, five articles were selected and studied in order to provide an instructional basis for future literature review as well as look for indications of an existing conceptual framework for SA. These articles were selected from Google scholar from among the top of the list returned for a search of ‘sentiment analysis’.

1. [64] is entitled “Sentiment Analysis: An Overview” and the author has written a thorough introduction to sentiment analysis and how to identify and extract subjective and opinionated information from text. It discusses various techniques and methodologies used in SA, including machine learning approaches, lexicon-based methods, and hybrid techniques. It also explores the applications of SA in areas such as social media monitoring, market analysis, and opinion mining. Lastly, the document addresses the challenges faced in SA, such as dealing with sarcasm, irony, and context-dependent sentiments.
2. [41] is entitled "Sentiment Analysis: A Combined Approach" and is published in the *Journal of Informetrics* in 2009 where it explores a hybrid method for SA. This approach integrates rule-based classification, supervised learning, and machine learning techniques. The authors evaluated their method on various datasets, including movie reviews, product reviews, and social media comments, and their findings indicate that combining multiple classifiers can enhance the effectiveness of sentiment analysis, achieving better precision and recall compared to using individual classifiers singularly.
3. [53] and provides a comprehensive overview of sentiment analysis from a linguistic perspective. It discusses how sentiment, characterized as positive or negative evaluation expressed through language, is analyzed. The text highlights common applications, such as determining the sentiment of online reviews for movies, books, or products. It also explores how SA is used in social media analysis by companies, marketers, and political analysts. The research focuses on extracting information from positive and negative words, the context of those words, and the linguistic structure of the text.
4. [15] is entitled "A Survey on Sentiment Analysis Challenges" is published in the *Journal of King Saud University - Engineering Sciences*, provides an examination of the various challenges faced in SA. It discusses issues such as the complexity of human emotions, the difficulty in detecting sarcasm and irony, and the challenges posed by multilingual SA. It also explores the limitations of current methodologies and the need for more advanced techniques to improve accuracy and reliability.
5. [62] is entitled "A Survey on Sentiment Analysis Methods, Applications, and Challenges," and provides a comprehensive overview of sentiment analysis. It discusses various methods used in sentiment analysis, including machine learning, lexicon-based, and hybrid approaches. The article also explores the applications of sentiment analysis in different domains such as social media, healthcare, and market research. Finally, it highlights the challenges faced in sentiment analysis, such as handling sarcasm, irony, and language-

specific issues. The authors aim to provide a complete understanding of the current state of sentiment analysis and suggest future research directions.

1.4.2 Bing Liu and *Sentiment Analysis*

Having read the aforementioned articles, and conducting additional research online, the author initially selected *Sentiment Analysis* by Bing Liu [28] as a framework because of references to the work, and the researcher's proximity and possibility as a mentor upon admission to UIC as a PhD candidate. It is not possible to summarize the entirety of Bing Liu's framework whose book runs over 360 pages. It is possible, however, to pick out five key topics within the work that help to characterize the problem space and provide a brief summary as each section is theoretically a domain in and of itself. For instance, at the beginning of the book, Liu discusses the intricacies of sentiment and opinion, feeling and emotions, facts and opinions, etc. Emotion itself is a complicated topic studied in an interdisciplinary fashion and an informative introduction can be found in [45]. The following are important overviews of concepts in Liu's book that provide a basis for understanding SA as an activity:

1. **Sentiment and Opinion – An overview of key concepts used in SA.**
2. **Aspect Classification – A look at a particular type of SA focused on smaller semantic units at the token level.**
3. **Entity Extraction, Search, and Linking – The foundations of generating ontologies automatically.**
4. **Sentiment Lexicon Generation – Building lexemic resources for analysis.**
5. **Mining Intentions – Anticipating the needs of a user for proactive activity.**

1.4.2.1 Sentiment and Opinion

> **Some who have read the book, or at any rate have reviewed it, have found it boring, absurd, or contemptible; and I have no cause to complain, since I have similar opinions of their works, or of the kinds of writing that they evidently prefer.**
>
> **J. R. R. Tolkein, *The Lord of the Rings***

Liu offers important definitions including sentiment, affect, emotion, mood, opinion, and subjectivity. *Sentiment* refers to the emotional tone expressed in a piece of text. SA typically categorizes sentiment as positive, negative, or neutral, known as a text's polarity. For example, a review saying "I love this holy hand grenade" expresses a positive polarity, while "I hate this silly English kinnigit" expresses a negative sentiment.

An *opinion* is a subjective statement that reflects personal beliefs, feelings, or thoughts about a particular subject and are often expressed in reviews, social media posts, and other forms of user-generated content. For instance, "it feels to me that I'm being oppressed" delineates a subjective interpretation of how politics during the Middle Ages affects the local peasant. As for *subjectivity*, it refers to the degree to which a text expresses personal feelings, views, or beliefs, as opposed to objective facts. A subjective statement is one that is influenced by personal feelings or opinions but is not identical to them, such as "I think this Monty Python and Holy Grail is fantastic." In contrast, an objective statement is fact-based and unbiased, like "The movie was released in 1975".

It is important to note the distinction. An opinion is a specific claim that reflects personal feelings, beliefs, or thoughts about a subject. It is an agent's viewpoint or value-based judgement, often made with assertoric force. For example, "I think John Cleese is hilarious" is an opinion because it conveys a judgment about a comedian who others find distasteful. In contradistinction, subjectivity refers to the wider concept of how personal feelings, perspectives, values, and biases influence one's worldview. It encompasses opinions but also includes the additional facets of the overall personal and emotional context in which they are formed. It is affected by the personal influence in any statement or perception. Describing King Arthur as "noble" is subjective because it reflects the individual's emotional valence towards Mr. Pendragon.

Additionally, terms such as feeling, emotion, affect, and mood are discussed. *Feelings* are the subjective instances and experiences of emotions. They are personal and private experiences that arise from varied emotions. For example, feeling happy about Monty Python is how you internally experience the emotion of joy which is understood more as abstracted description of those private experiences. On the other hand, *emotions* are intense, ephemeral responses to specific stimuli and perceptions. They involve physiological arousal, behaviors of expression, and conscious experiences. Examples fear of dangerous rabbits. They are often immediate and can be observed in the third person through facial expressions, body language, and linguistic claims about internal state. *Affect* is a broader term that encompasses the experience of feeling or emotion. It refers to the observables of emotions and can be seen by the tone of voice, facial expressions, and body language. It is construed as immediate and observable in response to stimuli. Lastly, mood is less intense but far longer-lasting states of mind. They are diffuse and seen as not tied to specific events. They influence one's overall emotional state and can last for hours to months. Examples include feeling calm when cows are being dropped or flung on you by Frenchmen with bad attitudes.

While the characterizations above are psychological in nature Liu provides definitions, methods, and descriptions pertaining to how one can determine these of a user given natural language corpora. He provides extensive conceptual analysis, methods, and reviews of literature for determining sentiment at the various levels of analysis: the document, sentential, and lexemic levels. He focuses on how entities and their aspects can be inferred from text. This is the thrust of SA: to determine these sorts of semantics based on lexicons, syntactic features, and other strategies. From a natural language perspective, it also provides a technical vocabulary for understanding what natural language ontology might be in computational terms.

1.4.2.2 Aspect Classification

> **Quality is a term that is used in many senses. One sort of quality is the differentia of substance, another is the capacity of doing or suffering, another is the passive quality, and another is the figure and the shape.**
>
> **Aristotle, *Categories***

Among the document, sentential, and token levels of classification, aspect classification is the most approachable since words and phrases often have simple connotations and are easier to parse than more complex textual structures. Liu devotes two chapters (5 and 6) to the topic and immediately singles out two important tasks: aspect extraction and aspect sentiment classification [28]. What is important to understand is that an aspect of an entity is a sub-part of an ontological primitive which is a mereological concept. For instance, in a review of a cell phone, he talks about how sound quality is not general, but is an aspect or feature of a cell phone. Connecting the dots, therefore, between the tokens ‘sound quality’ and ‘cell phone’ becomes a computational challenge.

Therefore, in a text, when an entity is identified, it is important not to conflate language that applies to aspects of an entity with the entity itself. Consider that one can make a comment of negative sentiment or polarity about an aspect of an entity, but have a positive sentiment about the entity as a whole. This differentiation between part and whole is quite simple and intuitional for a human, but in the semantic analysis conducted by an NLP system, can be challenging given the diversity of constructs available to the user of natural language.

He covers both supervised learning strategies and the use of lexicons to cope with the challenge, and then evolves the explanation into a set of sentiment rules, which is an open and extensible, domain-specific set of formal rules for achieving the same goals. He avers that the last of the strategies is the most complicated given the combinational explosion inherent in context-sensitive grammars of natural languages.

1.4.2.3 Entity Extraction, Linking, and Searching

> **We are prone to talk and think of objects. Physical objects are the obvious illustration when the illustrative mood is upon us, but there are also all the abstract objects, or so there purport to be: the states and qualities, numbers, attributes, classes. We persist in breaking reality down somehow… to be referred to by singular and general terms.**
>
> **W. V. Quine, *Ontological Relativity and Other Essays***

Identifying entities as targets of sentiment was introduced early in the text, and later on Liu goes to describe the problem space as dominated by four strategies: nomial frequency, syntactic relations of opinion/target and entity-part relations, supervised learning, and topic models. [28]. He also introduces a topic that eventually becomes the focus of his academic research: lifelong topic models. The main idea is that such models continuously grow and conduct knowledge-based topic modeling.

With Section 6.7.1, Liu introduces entity extraction as a problem closely aligned with named entity recognition (NER).[24] While he does not cover NER other than mentioning how important the problem space is in "information retrieval, text mining, data mining, machine learning, and NLP under the name of information extraction" [28], Liu goes on to give a problem statement:

> **Problem statement 6.1: Given a corpus *C*, we want to solve the following two subproblems:**
>
> 1. **Identify all entity expressions or mentions *M* in corpus *C*.**
> 2. **Cluster all entity expression in *M* into synonymous groups. Each group represents a unique real-world object or entity.**

Arguably, this is an instantiation of the classic notion of the bundle theory by David Hume.[25] From a mereological perspective, by gathering aspects into groups, it is possible to build out semantic models of entities. Speculatively entities subsumed by other entities are meronyms expressible as parent-child object relations and attributes assigned to entities are properties of objects.

Starting with Section 6.7.2, Liu begins explication of entity extraction claiming both hidden Markov models and conditional random fields, two supervised learning methods discussed earlier in the chapter in the context of detecting implicit aspects of entities. He continues rapidly to explain entity

[24] For a quick introduction consider [https://en.wikipedia.org/wiki/Named-entity_recognition].

[25] For a quick introduction consider [https://en.wikipedia.org/wiki/Bundle_theory].

linking in Section 6.7.3 providing a formal definition which on analysis seems to suggest hypernymy as the nature of the relationship. He again emphasizes the importance of keeping aspects and entities correctly aligned but distinct.

In the last section, 6.7.3. discussing entities search and linking, he discusses the problems polysemy and synonymy create and how important it is to resolve them. He considers them solvable as part of a classification model; he later discusses specific ML methods.

Entity linking has great potential to provide a metaphysical grounding for explaining natural language ontology. Given the complexity of the materials, the author advocates additional research. The topic, however, is undisputably metaphysical in nature, offering mechanics for addressing traditional meta-ontological disputes the analysis of which is beyond the scope of this paper.

1.4.2.4 Sentiment Lexicon Generation

> **Words, words, words.**
>
> **William Shakespeare, *Hamlet***

While natural language participates in conducting meaning from the morpheme up to the context and beyond into implicature, lexemes and simple phrases are the workhorses given the Principle of Compositionality. This principle is the idea that language is modular [52]. Therefore, the use of lexicons as a basis for determining sentiment is an extremely powerful tool, greater context notwithstanding. Liu discusses two important strategies.

The first strategy is the *dictionary-based approach* [28, p.190]. Starting with a small set of human-labeled seed words, it becomes possible to use various methods, such as probabilistic algorithms, to grow the lexicon with the polarity mappings. Usually, this includes a positive-, negative-, and neutral-oriented seed set. Then, through the use of classification and other lexical resources such as semantic frames, it is possible to grow the sentiment lexicon.

The other strategy offered is the *corpus-based approach* [28, p.193]. With this option, seed words are either begun within a semantic domain or have evolved from a general lexicon into a domain-specific set. The latter approach introduces difficulties since tokens are subject to polysemy across multiple domains when transformed from a general lexicon. If the corpus is large and complex, then the first strategy is the better choice since lexicon adaptation is a challenging task.

1.4.2.5 Mining Intentions

> **Intention appears to be something that we can express, but which brutes… can *have*, though lacking any distinct expression of intention.**
>
> **G.E.M. Anscombe, *Intention***

Liu devotes a short, but complete chapter to intention mining [28, p.250]. In it, he outlines a distinction between intention and sentiment. He immediately offers a dictionary definition:

> **Definition 1.1 (Intention): *Intention* has two main meanings or senses:**

1. **A course of action that a person or a group of persons intends to follow.**
2. **The goal or purpose behind a specific action or set of actions.**

Using this definition as a basis, he discusses nuances of intention in language. He also exemplifies what he terms “pseudo” intentions.

The second section of the chapter discusses intention classification. After providing a brief survey of techniques, he goes on to the next section discussing “fine-grained mining” of intention. He admits such methods which attempt to tackle additional semantic granularity have “not been tested or validated using real-life experiments” [28, p.257].

1.4.2.6 Precising Definitions and Tasks

Having laid the foundation for a conceptual framework by selecting important concepts and applications of SA, the following definitions and tasks as provided by Bing Liu provide a more focused technical vocabulary for communicating and researching SA. These represent the highlights of a collection of precising definitions and task specifications that serve as a model for building out a versatile SA system. The Kaggle Sentiment Analysis Dataset presents a very limited aspect of SA, and to broaden the exploration beyond a suite of Tweets and simple mechanisms for establishing positive, negative, or neutral polarity, the following definitions and tasks empower the SA researcher to attack the problem with more sophisticated intellectual tools.

Given the key concepts covered in Section 1.4.2, a complete inventory of precising definitions provided in the second chapter is presented and makes manifest the range of concepts:

Opinion Holder
Time of Opinion
Definition 2.1 Opinion Quadruple
Definition 2.2 Sentiment Target
Definition 2.3 Entity
Definition 2.4 Sentiment
Definition 2.5 Rational Sentiment
Definition 2.6 Emotional Sentiment
Sentiment Orientation, Intensity, and Rating
Definition 2.7 Opinion Quintuple
Definition 2.8 Opinion Reason
Definition 2.9 Opinion Qualifier
Objective of Sentiment Analysis
Key Tasks of Sentiment Analysis
Definition 2.10 Entity Category and Expression
Definition 2.11 Aspect Category and Aspect Expression
Definition 2.12 Explicit Aspect Expression
Definition 2.13 Implicit Aspect Expression
Model of Entity
Model of Opinion Document
Definition 2.14 Aspect-Based Opinion Summary
Regular Opinion
Comparative Opinion
Subjective Opinion
Fact-Implied Opinion

Besides definitions, Liu also provides a general process for conducting SA at the document level in 8 specific tasks [28, p. 27].

Task 1 – Entity extraction and resolution
Task 2 – Aspect extraction and resolution

Task 3 – Opinion holder extraction and resolution
Task 4 – Time extraction and standardization
Task 5 – Aspect sentiment classification or regression
Task 6 – Opinion quintuple generation
Task 7 – Opinion reason extraction and resolution
Task 8 – Opinion qualifier extraction and resolution

It is on the foundation of terminology explored briefly in Sections 1.4.2.1 to 1.4.2.6 that Liu's framework rests. This paper does not review further chapters in *Sentiment Analysis* in this paper which would extend conceptual summaries to other useful strategies and methods including document-level and sentence-level sentiment analysis.

2 The Kaggle Sentiment Analysis Dataset[26]

While theory is indispensable to exploratory research, empirical practices compel the researcher to build software to determine constraints and to use exemplars of practice to inform theory. In this section, the Kaggle Sentiment Analysis Dataset project serves as a real-world example meant to meet the requirements of the exploratory research set out in Section 1.2 and included in the scope of research methodologies listed in Section 1.3.1. The Dataset provided was a data source extracted from tweets from a variety of users across the globe. The documentation on the Data Card tab is brief and the paper presents it in its entirety here as a blockquote:

> **About Dataset**
>
> **Context**
>
> **There's a story behind every dataset and here's your opportunity to share yours.**
> **training data was automatically created, as opposed to having humans manual [sic] annotate tweets. In our approach, we assume that any tweet with positive emoticons, like :), were positive, and tweets with negative emoticons, like :(, were negative.**
> **Content**
>
> **What's inside is more than just rows and columns. Make it easy for others to get started by describing how you acquired the data and what time period it represents, too.**
> **The data is a CSV with emoticons removed. Data file format has 6 fields:**
>
> > **0 - the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)**
> > **1 - the id of the tweet (2087)**
> > **2 - the date of the tweet (Sat May 16 23:58:44 UTC 2009)**
> > **3 - the query (lyx). If there is no query, then this value is NO_QUERY.**
> > **4 - the user that tweeted (robotickilldozr)**
> > **5 - the text of the tweet (Lyx is cool)**
> > **Acknowledgements**
>
> **We wouldn't be here without the help of others. If you owe any attributions or thanks, include them here along with any citations of past research.**
> **Thanks kaggle team for inspiring me.**
>
> **Inspiration**
>
> **Your data will be in front of the world's largest data science community. What questions do you want to see answered?**
> **This dataset is for world data scientists to explore experiments in sentiment analysis.**

2.1 Observations from Reviewing the Dataset

Since this phase of the research used empirical methods, the first step was to conduct observations of the Dataset and solutions. A strategy for conducting analysis quickly coalesced from the inability to review the entire volume of more than fifty solutions available on the site. By selecting 20% of the solutions and ranking them by rating (votes for popularity), it was easy to discard the lower quality solutions. Once the solution set had been selected and manually inspected, the strategy was to first

[26] The Dataset is located at [https://www.kaggle.com/datasets/abhi8923shriv/sentiment-analysis-dataset].

decide which libraries were used, and then to decide on a more specific strategy motivated not by producing a solution that had a high performance rate, but rather by seeing how the rather arbitrary of formal methods affected the use of ML models.

2.1.1 Sample Solution Selection

After reviewing the entirety of the Dataset, it first became clear that the site contained little useful information or discussion, however, besides the data itself, over fifty solutions were available at the commencement of the research. All of the datasets were Python-based solutions, and because it was possible to list them by popular ranking, it was easy to produce a list of ten Jupyter notebooks that were demonstrative of a set of solutions for modeling:

Author	**Folder**	**URL**
Mohsin Saleem	1_sentiment-analysis-by-nlp_mohsinsial	https://www.kaggle.com/code/mohsinsial/sentiment-analysis-by-nlp
Pooja Gupta	2_sentiment-analysis-machine-learning-approach_poojag718	https://www.kaggle.com/code/poojag718/sentiment-analysis-machine-learning-approach
neural-net-rahul	3_sentiment-analysis-using-rnn_rahulabrsl	https://www.kaggle.com/code/rahulabrsl/sentiment-analysis-using-rnn
Ammar Al-Mekhlafi	4_sentiment-analysis-using-xlstm_ammaralmekhlafi	https://www.kaggle.com/code/ammaralmekhlafi/sentiment-analysis-using-xlstm
Harsh Der	5_sentiment-analysis_hder03	https://www.kaggle.com/code/hder03/sentiment-analysis
Hakim11	6_deep-text-analysis-with-lstm-and-keras-tuner_hakim11	https://www.kaggle.com/code/hakim11/deep-text-analysis-with-lstm-and-keras-tuner
Zeyad Sayed	7_sentiment-analysis-using-twitter-dataset_zeyadsayedadbullah	https://www.kaggle.com/code/zeyadsayedadbullah/sentiment-analysis-using-twitter-dataset
Md. Reyad Hossain	8_sentiment-analysis-model-applying_mdreyadhossainnsu	https://www.kaggle.com/code/mdreyadhossainnsu/sentiment-analysis-model-applying
Akanksha Srivastava	9_sentiment-analysis-dataset_akanksha10	https://www.kaggle.com/code/akanksha10/sentiment-analysis-dataset
Ayush Batra	10_sentiment-analysisss_ayushbatra	https://www.kaggle.com/code/ayushbatra/sentiment-analysisss

Table 1: Sample Selection of Solutions for Analysis

2.1.2 Solution Library Reviews

To determine the best way to craft a solution for this challenge, the author reviewed source code for the ten solutions above. One of the first observations available was that the various authors chose a diverse and different collection of libraries. After documenting and classifying the import statements for each of the solutions, the following table was derived to represent a collated version of the individual import lists for review:

Import Statement and Import Classification
import warnings (core)
import os (core)
import re (core)
import string (core lib)
from string import punctuation (core lib)
import numpy as np (data/ML)
import pandas as pd (data/ML)
import seaborn as sns (visualization)
import matplotlib.pyplot as plt (visualization)
from keras import Sequential (keras)
from keras.layers import Dense,SimpleRNN,Embedding,Flatten (keras)
from keras.preprocessing.text import Tokenizer (keras)

from keras.utils import pad_sequences (keras)
from keras.utils import to_categorical (keras)
import nltk (nltk)
From nltk.stem import LancasterStemmer (nltk)
from nltk.stem.wordnet import WordNetLemmatizer (nltk)
from nltk.tokenize import word_tokenize (nltk)
from scipy.sparse import hstack (scipy)
from sklearn.ensemble import RandomForestClassifier (scikit-learn)
from sklearn.feature_extraction.text import CountVectorizer (scikit-learn)
from sklearn.feature_extraction.text import TfidfVectorizer (scikit-learn)
from sklearn.linear_model import LogisticRegression (scikit-learn)
from sklearn.metrics import accuracy_score (scikit-learn)
from sklearn.metrics import accuracy_score,classification_report, ConfusionMatrixDisplay (scikit-learn)
from sklearn.metrics import classification_report (scikit-learn)
from sklearn.model_selection import train_test_split (scikit-learn)
from sklearn.preprocessing import LabelEncoder (scikit-learn)
from sklearn.preprocessing import OneHotEncoder, StandardScaler (scikit-learn)
from sklearn.tree import DecisionTreeClassifier (scikit-learn)
from spacy import load (spacy)
import tensorflow as tf (tensorflow)
from tensorflow.keras import layers (tensorflow)
from tensorflow.keras.layers import Input, LSTM, Dense, Embedding, concatenate (tensorflow)
from tensorflow.keras.models import Model (tensorflow)
from tensorflow.keras.preprocessing.sequence import pad_sequences (tensorflow)
from tensorflow.keras.preprocessing.text import Tokenizer (tensorflow)

Table 2: Sample Selection of Solutions for Analysis

Given the sheer volume of different libraries and the lack of familiarity with the various API collections, the author divided the list into two sets: accepted and rejected. The accepted set was the best guess on which assortments of functionality would advance the research solution. The following is the table:

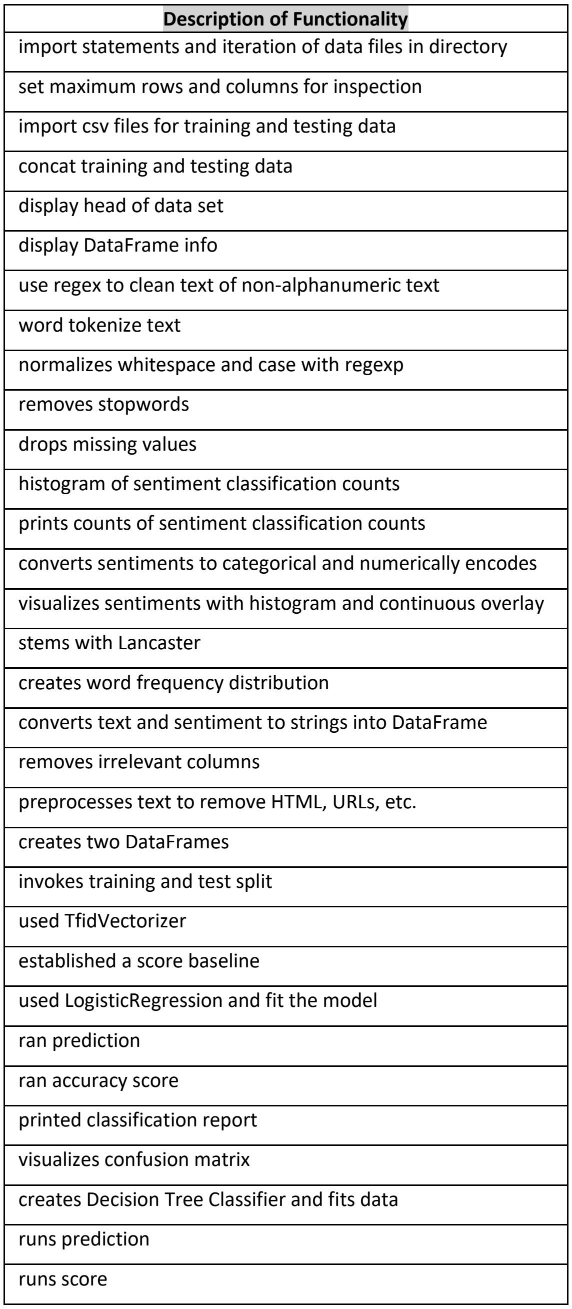
Library Name (Acceptance or Rejection)
standard libraries (accept)
matplotlib (accept)
pandas (accept)
seaborn (accept)
sklearn (accept)
nltk (accept)
numpy (accept)
keras (reject)
spicy (reject)
spacy (reject)
tensorflow (reject)

Table 3: Preliminary Import List

The primary criterion for rejection was the amount of time it would take to review a resource. Ultimately, a review of the Python functionality [29], Pandas and NumPy functionality [34], Python ML methods (Müller & Guido, 2016), and Python NLP methods [3] suggested that besides the standard and visualization library sets, Pandas, NumPy, Scikit-Learn would be sufficient in order to craft a solution comparable to the solution set. Note, the custom solution did not require Seaborn and NumPy since the former builds upon the basic functionality of Matplotlib, and the latter is better applied to data science rather than ML solutions.

2.1.3 Solution Inventory and Development

Having selected a general class of libraries, it became necessary to analyze the implementation functionality of the notebooks. That involved reviewing and labeling the functionality of the of each offering of the model solution set. As an example, here is the functionality of the solution deemed closest to the minimum requirements selected in Section 2.1.2:



Description of Functionality
import statements and iteration of data files in directory
set maximum rows and columns for inspection
import csv files for training and testing data
concat training and testing data
display head of data set
display DataFrame info
use regex to clean text of non-alphanumeric text
word tokenize text
normalizes whitespace and case with regexp
removes stopwords
drops missing values
histogram of sentiment classification counts
prints counts of sentiment classification counts
converts sentiments to categorical and numerically encodes
visualizes sentiments with histogram and continuous overlay
stems with Lancaster
creates word frequency distribution
converts text and sentiment to strings into DataFrame
removes irrelevant columns
preprocesses text to remove HTML, URLs, etc.
creates two DataFrames
invokes training and test split
used TfidVectorizer
established a score baseline
used LogisticRegression and fit the model
ran prediction
ran accuracy score
printed classification report
visualizes confusion matrix
creates Decision Tree Classifier and fits data
runs prediction
runs score

printed classification report
visualizes confusion matrix
creates RandomForestClassifier and fits model
runs prediction
runs score
printed classification report
visualizes confusion matrix
prints three scores for comparison
defines a manual test

Table 3: Sample Solution Description

Given this Dataset solution as a template, the author developed and executed a custom solution which is listed in Appendix A. It is important to again emphasize that this exploratory research does not attempt to optimize the ML solution as the author simply does not have the expertise in ML to attempt such a task. Rather, the custom solution is an attempt to establish a general process for building a hybrid SA solution that uses both formal NLP and statistical ML methods and to accrue a general familiarity with resources to foster that ability in further research.

In the author's opinion, merely proving out that one has ML and Jupyter skills is less important in conducting SA research than spreading research methodologies out across the variety listed in Section 1.3.1. In other words, a researcher should balance the design and implementation of an ML solution against the seven other research methodologies listed to ensure that at the conclusion of the research, the research product has identified a wide range of relevant research issues, explanatory gaps, terminology, and resources. After such exploratory research, it then makes sense to accrue the skill set to conduct more intensive research on the aspect of Python-based implementations of SA systems since nothing constrains software architecture specifically to Python despite the language's strengths and widespread currency in academia and industry.

2.1.4 Brief Description of Custom Solution Functionality

The application organizes the various blocks of code four phases to separate concerns:

2.1.4.1 Initial Processing

First, in the 'Utility Imports' block, the application imports all functions at the top of the file, and those statements construct the imports as narrowly as possible. The primary functionality for the application is from the NLTK and Scikit-Learn libraries. The application implemented aliases as a convention to make the imports in the source code easy to identify by prefixing with the respective, all-caps header name such as NLTK_ and SKLEARN_. In the 'Global Data' block, after, the code has a series of global data declarations for the application again using the uppercase convention to set off the tokens further on for easy identification. Next, in the 'Training and Testing Data both data types are imported in the hand Dframe data structure and irrelevant columns are dropped as well as some minor conditioning to manage NaNs.

2.1.4.2 Formal NLP

The Kaggle data is preprocessed using regex to remove HTML, URLs, line breaks, punctuation, and excess whitespace. Tokenization of the string follows with the removal of stop words which are high frequency words like 'of' and 'a'. Lancaster stemming follows converting words into base forms by dropping conjugations of verbs and declensions of nouns, etc. And then the memory usage is examined, and the data is probed and visualized to examine the polarities provided. These are the formal methods, and the author did not select them with a specific strategy in mind, but rather as proof of concept based on the ease of implementation in the construction of the custom solution whose purpose is merely proof of concept.

2.1.4.3 ML Processing

Next, the application builds four models with the first for establishing a baseline: train and test split, logistic regression, decision tree classifier, and random forest classifier each being followed by reporting and visualization with a confusion matrix display. The author chose these three models based on a review of criteria given in the descriptions in (Müller and Guido, 2016). It was not the author's intent to select an optimal model, but to mix and match the formal NLP methods in Section 2.1.4.2 with the ML versions in this section to simply see what results from a somewhat arbitrary mixture of strategies and to have source code to compare against Liu's framework.

2.1.4.4 Cross-Model Evaluation and Manual Testing and Validation

Lastly, the research compared the models with and with and without pairing the ML with the formal NLP strategies. This experiment was to affirm the intuition that without a carefully reasoned set of strategies of using formal NLTK methods, the performance of the solution would suffer. Lastly, the research demonstrates proof of concept for manual testing at the end with a few custom strings that express simple sentiment.

2.1.5 Scoring

This research conducted a simple analysis of scoring for a simple case of cross-validation for several models:

1. **Baseline model**
2. **Logistic regression**
3. **Decision Tree Classification**
4. **Random Forest Classifier**

Before explaining the broader strategy of scoring, a brief review of the four models based on Chapter 2: Supervised Models on page 27 of [36] above is warranted.

2.1.5.1 Baseline (Split and Test)

The split and test method is known for its simplicity and ease of implementation. It helps prevent overfitting by evaluating the model's performance on unseen data, ensuring it can generalize well to new data. However, the performance can vary significantly depending on how the data is split, and with small datasets, splitting can lead to insufficient training data, affecting model performance. Additionally, if the split is not representative, it can introduce bias into the evaluation.

2.1.5.2 Logistic Regression

Logistic regression is highly interpretable, providing clear insights into the importance of features through coefficients. It is efficient, fast to train, and works well with large datasets. The model outputs well-calibrated probabilities, making it useful for binary classification tasks. However, it assumes a linear relationship between the input features and the log-odds of the outcome, which may not always hold. Logistic regression can be sensitive to outliers, which can skew the results, and it is prone to overfitting with high-dimensional data if not regularized.

2.1.5.3 Decision Tree Classifier

Decision tree classifiers are easy to understand and visualize, making them accessible for non-technical stakeholders. They can handle both numerical and categorical data and capture non-linear relationships. Additionally, they require minimal data preprocessing, such as normalization or scaling. However, decision trees are prone to overfitting, especially with small datasets or deep trees. They can be unstable, with small changes in the data leading to significantly different trees, and can create biased trees if some classes dominate.

2.1.5.4 Random Forest Classifier

Random forest classifiers generally provide high accuracy and robustness by averaging multiple decision trees, making them less prone to overfitting compared to individual decision trees. They also provide insights into feature importance, aiding in feature selection. However, random forests are computationally intensive, requiring more resources and time for training and prediction. They are more complex and harder to interpret than individual decision trees and may struggle with imbalanced datasets, requiring additional techniques to address this.

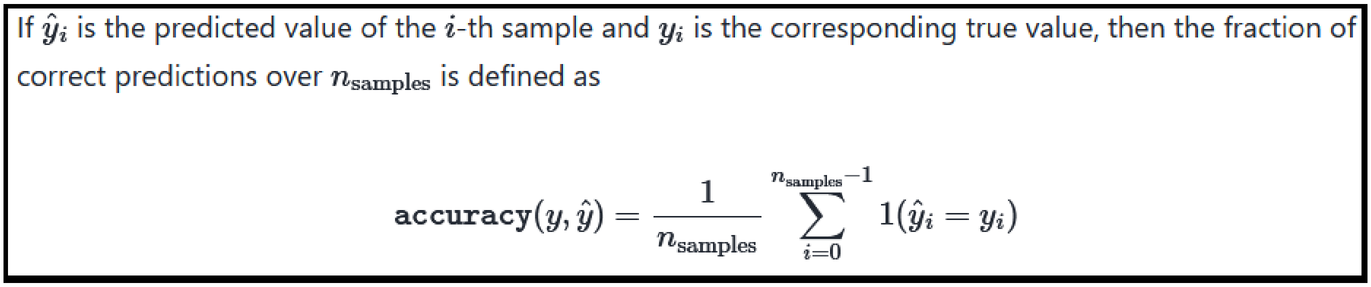
2.1.5.5 Scoring Description

The process begins with splitting the data into training and testing sets for both the Lancaster and the default Kaggle datasets. This ensures that the models can be trained on one portion of the data and evaluated on another, providing a measure of their performance on unseen data. Next, the text data is converted into numerical format using TF-IDF vectorization. This step transforms the textual information into a format that machine learning models can process. The baseline model is then established by calculating the baseline accuracy, which is determined by taking the most frequent class in the dataset. This provides a reference point against which the performance of more complex models can be compared. A logistic regression model is trained, and its accuracy is evaluated on the test set. The results are further detailed through classification reports and confusion matrices, which provide insights into the model's performance across different classes. Similarly, a decision tree classifier is trained and evaluated.

The accuracy of these models is also assessed, and classification reports and confusion matrices are generated to understand its performance. The random forest classifier, an ensemble method, is then trained and evaluated. Like the previous models, its accuracy is measured and detailed through classification reports and confusion matrices. Finally, a cross-model evaluation is conducted to compare the performance of the different models (baseline, logistic regression, decision tree, and random forest) on both the Lancaster and Kaggle default datasets. The evaluation scores (also known as accuracy scores) for each model are printed, allowing for a comprehensive comparison of their effectiveness.

The accuracy score is defined as the ratio of correctly predicted samples to the total number of samples. It is calculated using the accuracy_score function from the Scikit-Learn metrics module. This function takes the true labels and the predicted labels as inputs and returns the accuracy as a float value. The accuracy score is a simple and intuitive metric for evaluating classification models, especially when the classes are balanced, and in this case, by comparing across 4 different models in two modes, we can see the variance that is inherent in using different techniques, although a full analysis of the mathematics beyond some simple formulas is not possible given the author's lack of ML training.

What is important to know is that the analysis begins with a mathematical indictor function. A mathematical indicator function, also known as a characteristic function, is a function that indicates membership of an element in a set, which is also known as an extension or a comprehension when given in set builder notation. It functions by mapping all inclusions to unity, and all exclusions to zero. In the accuracy_score function used, 1(x) is the indicator function. According to the documentation, "In multilabel classification, the function returns the subset accuracy. If the entire set of predicted labels for a sample strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise, it is 0.0." The formula used by the system can be seen in Figure 1.



If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value, then the fraction of correct predictions over n_{samples} is defined as

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

Figure 1: Accuracy Function Used to Assess Polarity

Thus, in SA, particularly when assessing the polarity of text (i.e., determining whether the sentiment is positive, negative, or neutral), an indicator function can be used to evaluate the accuracy of the model's predictions. For each text sample where the indicator function is defined, where *y_i* is the true sentiment label (e.g., positive, negative, neutral) and *$\hat{y}_i$* is the predicted sentiment label. The indicator function takes the value 1 if the predicted sentiment *$\hat{y}_i$* matches the true sentiment *y_i*, and 0 if it does not. The accuracy score is then calculated as the mean of the indicator function values across all text samples where *n* is the total number of text samples. In summary, the indicator function helps determine whether each individual prediction is correct (1) or incorrect (0). By averaging these values, you obtain the overall accuracy of the sentiment analysis model.

2.1.5.6 Results

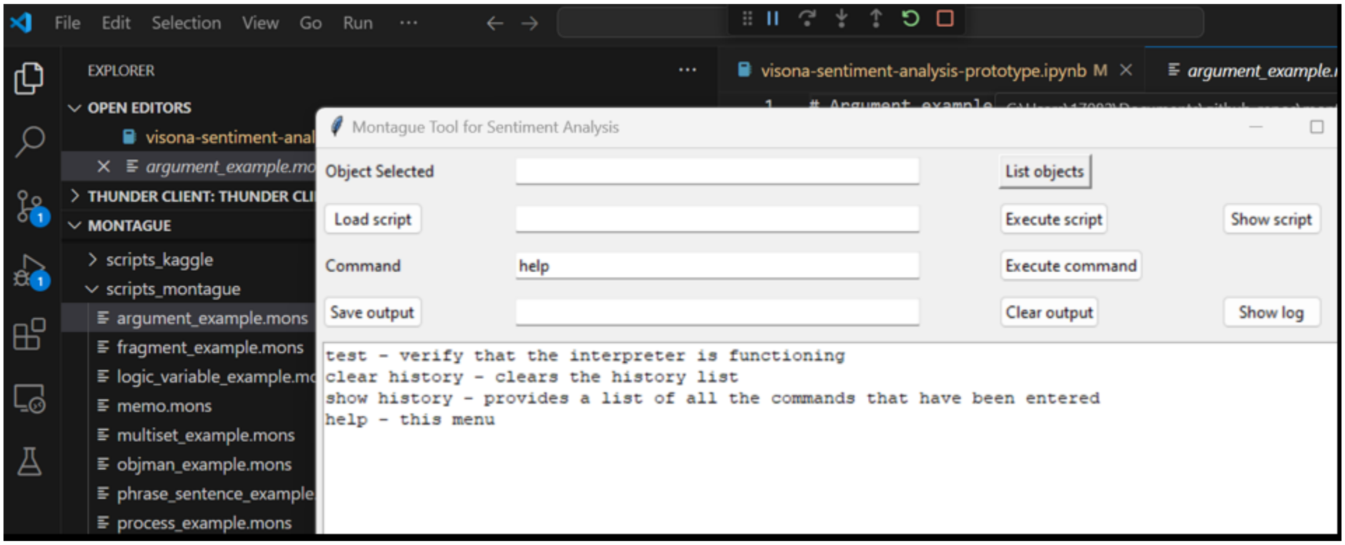
Without NLP	
Model	Score
Baseline Model	40.5%
Logistic Regression	68.5%
Decision Tree Classification	61.8%
Random Forest Classifier	68.1%
With NLP	
Model	Score
Baseline Model	40.5%
Logistic Regression	83.0%
Decision Tree Classification	76.0%
Random Forest Classifier	81.3%

Table 4: Custom Solution Results

As predicted, using NLTK without a strategy of an analysis of text simply decreased the effectiveness of the ML methods. The reasoning for this is simple; without a coherent application of functions with an awareness of the nature of the text itself, the functions simply altered the text in a way that impacts the signature of the data from the initial training offered by the Dataset. As this fundamentally changes the distributional semantics, the learning models are dealing with a subtly altered corpus, and classification based on training becomes less effective. This highlights the importance of being familiar with the nature of the text in the corpora when selecting which NLTK functions to apply to the data. It therefore remains an open question on how to use NLTK optimally with models to improve performance

3 Montague: Formal System Object Management System[27]

Using Python-based NLP and ML libraries is a specific technical skillset and acquiring proficiency with Python for this project was time consuming. In order to gain proficiency in basic Python, software development commenced with implementing an object management system which is designed to speed up the development and encourage the reuse of Python solutions by encapsulation and automation of the creation and management of various objects through a set of flexible and varied classes that serve as a foundation to any SA project. One particular goal was to set up a system that would allow for the execution of arbitrary formal systems that might be relevant in mathematics and logic. NLP has a formal side that relies on grammatical manipulation of strings above the level of regular expressions. An additional goal was to provide an interface to grammars represented in the Grammatical Framework, though time allotted for research did not allow for development with this tool.[28]



Command: help

```
test - verify that the interpreter is functioning
clear history - clears the history list
show history - provides a list of all the commands that have been entered
help - this menu
```

Image 1: Montague Tool For Sentiment Analysis

The motivation for building an object management system is simple. By creating a base class and deriving sub-classes of arbitrary types, it becomes possible to implement the MVC design pattern as a preliminary architecture for programmatic control of resources. (See *Image 1*) Of particular importance is that the system implements logging, and allows for writing scripts to implement not just Jupyter notebooks, but arbitrary scripts called with the MONS (MONtague Script) extension for programmatic control across NLP tools. For instance, a Montague script could create an SA object which encapsulates the same functionality of the Jupyter notebook, use methods to customize the notebook by selecting various ML strategies, and programmatically alter the parameters of the invoked models systematically to conduct multiple instances of cross-validation. Consequently, additional methods can pipe those results to the DBMS or other integrated

[27] The GitHub repository is located at [https://github.com/jtvisona/montague] and at the time of this paper is in a preliminary stage of development.

[28] The Grammatical Framework pioneered by Arne Ranta can be found at [https://www.grammaticalframework.org/]. An introductory book for using the GF language is [43]. The author would like to integrate GF into future research.

applications. The ultimate goal would be the development of a 4th generation domain-specific language for working with SA challenges.

```
class ObjectManager( Object ):

    _object_list : dict = field( default_factory=dict )
    _linked_interpreter : object = field( default_factory=object )

    def __init__( self, name="", interpreter="" ):
        logger.debug( "Called " )
        if not isinstance( interpreter, INT.Interpreter ):
            message = "No interpreter to ObjectManager"
            logger.error( message ); raise Exception( message )
        # pass error check so log
        logger.debug( f"interpeter={type(interpreter)}" )

        self._object_list = []
        logger.info( "Assigning intepreter to object manager" )
        self._linked_interpreter = interpreter

    @property
    def object_list( self ):
        return self._object_list

    def add_object( self, obj: object ) -> bool:
        self._object_list[ obj.uuid ] = obj
```

Listing 1 – Object Manager Fragment

In *Listing 1* immediately above, the skeleton of an object manager class creates a framework for adding, linking, listing, etc. objects of an arbitrary sort. In future versions, for instance, any entity extraction performed in an SA class could in theory programmatically create objects of aspects, entities, documents, etc. to create Python instances for programmatic manipulation. See *Listing 2* to look at the Base Object class.

```
class Object:
    __uuid: str = ""
    __name: str = ""
    __delim: str = "`"

    def __init__ ( self, name: str = "" ):
        if G.debug:
                    print( f"{super().__class__}.{I.currentframe().f_code.co_name}() \
                       called by {I.stack()[1].function}()" )
            print( f"* Args:\n\tname='{name}'" )
        self.__uuid = uuid4() # returns UUID obj and not string; to convert 
str(self.__uuid)
        self.__name = name
```

Listing 2 – Base Object Class

In such an environment, discovered natural language structure can be implemented in the run-time to extend programmatic and meta-programmatic abilities of the researcher. Furthermore, the object manager could have object-relational mapping functionality built in. Lastly, note that such control over SA development would enjoy the benefit of custom exception handling and logging. This aspect of research is preliminary but offers proof of concept for the development of tools for future research.

4 Research Analysis, Synthesis, and Speculation

If we knew what we were doing it wouldn't be research, would it?

Albert Einstein

> **Research philosophy is primarily concerned with cognitive theory and its relevance to creating and expanding knowledge… in other words creating and expanding what we know about any aspect of the universe.**
>
> **Mbanaso et al., *Research Techniques for Computer Science, Information Systems and Cybersecurity***

4.1 Research Conclusions

Having conducted exploratory research in a manner described above, it is now incumbent upon the author to analyze, synthesize, and speculate given the acquisition of knowledge, no matter how preliminary the research findings are. The author advances the following five claims, and then supports them.

1. **SA is an extremely complicated discipline with competing frameworks and models, methods, etc.**
2. **SA is exceedingly useful in academia and industry for predicting polarity (positive, negative, neutral) of sentiment.**
3. **SA requires a firm grasp of both formal and computational NLP techniques.**
4. **SA is a very empirical process that requires researcher expertise and trial and error.**
5. **A software tool and domain-specific language to conduct future SA research is necessary.**

4.1.1 The Complexity of SA

SA is not an algorithm, or even a collection of algorithms and heuristics. It is an approach to semantic analysis that shadows the wider discipline of NLP. Thus, all of the complexity inherent in regular NLP inheres to SA. No one fully understands natural language, in fact, which is an extremely complicated phenomenon, and philosophers of linguistics and language have articulated hundreds of contentious theses.

There are readily demonstrable sources of complexity. Natural language is inherently ambiguous and vague. Words and phrases can have multiple meanings depending on the context. Understanding the meaning of a sentence often requires context. The structure of sentences can vary widely, and the same words arranged differently can convey different meanings. Parsing is a significant technical challenge, particularly when language is informal and unstructured and divorced from context. Other facts, such as cultural and social factors which might often be thought of as common sense by a natural language user, influences language use. And humans practicing semantic analysis rely heavily on pragmatics, which involves understanding the intended meaning without explicature relying instead on implicature.

4.1.2 The Utility of SA

SA is an extremely useful aspect of NLP. It helps businesses gauge how customers feel about their products, services, or brand and allows them to discover their reasoning and the language by which customers think. By analyzing social media posts, and other forms of linguistic feedback, corporations can identify areas on which to improve their products and services. SA also allows for real-time monitoring of public opinion. This is particularly valuable during marketing campaigns. SA, therefore, provides insights into market trends and preferences, and SA allows firms to improve their support, manage their reputation, and do research on opposition and competition.

4.1.3 The Dual Strategies of NLP

Successful practitioners of SA rely on both formal linguistics and computational methods when architecting software systems. Formal methods, such as rule-based systems, provide clear, interpretable rules and structures and dominate in scenarios where linguistic rules are well-understood and well-defined. On the other hand, statistical methods, including machine learning and deep learning, are powerful in handling large datasets and capturing patterns that are difficult to detect or describe. By combining both approaches enhances the robustness and can improve the accuracy of formal methods by providing probabilistic information that helps in disambiguating language. Since natural language is inherently ambiguous, statistical methods are particularly effective at leveraging enormous amounts of data to probabilistically predict the meanings.

4.1.4 The Importance of Researcher Expertise

SA requires a tremendous amount of human expertise. Human language is subtle, nuanced, and complex. The subtleties of sentiment, such as sarcasm, irony, and context-specific meanings, require advanced knowledge of linguistics and NLP methods. Preparing data for SA involves cleaning and formatting large datasets, which is more of an art than a science, and this step is crucial for ensuring accuracy. Different algorithms and models each have strengths and weaknesses, so selecting the right ones and fine-tuning specific applications and domains requires extensive knowledge and experience. SA must also account for the linguistic context which is often difficult to disambiguate. Lastly, as language evolves over time, experts need to continuously update and adapt the source code and methods.

4.1.5 The Necessity of a Framework for Development

Given all of the factors above and the enormous amount tools and resources that are required to practice SA, having a tool devoted to continuous improvement and testing of source code is necessary. Just in the python ecosystem there are more than half dozen libraries (NLTK, Spacy, Keras, PyTorch, Scikit-learn, etc.) that provide tools in building practical systems. The theory that underlies these tools is also highly mathematical in nature, and to that end, having a platform to ease the integration of the tools and data is beneficial.

4.2 Validating the Preliminary Research into the Research Problem

To review the research problem posed earlier in this paper:

> **Research Problem: Explore the nature of SA and related strategies in non-logical aspects of semantic analysis, by 1) understanding its relationship to NLP, AI, and computer science more broadly, 2) selecting and distilling important features of an SA theoretical framework, 3) describing a methodological philosophy and framework for accumulating SA knowledge, 4) acquiring practical knowledge of a simple SA problem, 5) developing a simple technical platform for future SA studies, and lastly 6) reviewing, analyzing, synthesizing, and speculating about the SA research performed and future strategies for the continuation of SA research at the graduate level.**

The author now maintains all six requirements to satisfy the claim that the exploratory research is effective have been met. First, an adequate characterization of SA has been determined after a description of the relationship between the topic and the greater domains to which it belongs, primarily NLP and AI. Next, this exploratory research has documented and explained the essential features of a theory to understand SA well. This research has also justified its philosophy and structure, and that theory has been strengthened by the analysis and development of a simple Python-based NLP hybrid solution. In these closing sections, the review and analysis serve as a basis for future research.

4.3 Speculation and Future Research

First, the author chose exploratory research composed of six distinct methodologies because prior to this project, the author had no experience in Python, ML, or SA. Coming to the project with a healthy understanding of linguistics, computer science, and philosophy, the primary goal was to move from high-level theory about language and computation towards a challenging aspect of NLP research. To this end, the exploration has exposed how important ML is to SA, NLP, and AI. One of the chief weaknesses in this research was the inability to use formal methods like those of NLTK and statistical and computational methods like Scikit-Learn in order to conduct more directed, purposeful experiments with big data. Thus, future education and research for the author will be in the theory and practice of ML in the form of continued work on Montague as well as the possible pursuit of a second master’s degree in data science and ML. The field is simply too deep and wide not to accumulate expertise should a real-world SA project be the outcome of this research.

Secondly, while mastery of the theory behind SA and NLP more broadly will happen over time from general resources, future research must also engage other frameworks and additional primary source literature. Given the limited scope of this investigation, too many resources were simply neglected on account of the need for brevity and to deliver findings within the four-month period which the author was allotted for his research and analysis. No determination, for instance, was made of which journals are contemporaneously authoritative in the field. The work of no other experts other than Liu was systematically investigated, nor were their frameworks examined and compared and contrasted with [28]. And the documentation for the various tools is vast and demands rigorous investigation.

Lastly, the Kaggle Sentiment Analysis Dataset is an overly simplistic, low-quality real-world example of SA and NLP. Even if the author improves both his ML skills and fortifies his theoretical knowledge of SA, large, complex, real-world projects introduce a host of concerns and problems not readily apparent to an academic conducting research. Social dimensions to SA exist that have not been considered, and thinking about financial and computational constraints have been ignored in this exploratory work. Therefore, it behooves the author to not only continue with Python development (both the Monague object management system and various ML toolkits), but to identify contacts and opportunities in the corporate sector to gather additional information about the ground-truths of implementing SA architectures given how many solutions seem to be proprietary trade secrets.

5 References

5.1 Background

References are provided in the standard APA format with the exception of articles in the Stanford Encyclopedia of Philosophy (SEP). Those are given in the format recommended by the editor. The author has attempted to stay within his personal library, and besides the notable academic presses such as Cambridge, Oxford, Chicago, and MIT presses, the author has leaned heavily on the SEP, O'Reilly and Associates, and Springer-Verlag for materials given their commitments to quality. Future research in this topic is likely to expand exploratory research to the Encyclopedia of Philosophy and Internet Encyclopedia of Philosophy as well as documentation from particular Python libraries. Given this is only a masters-level research and development project, it was important to draw boundaries for the sake of brevity.

The core knowledge gained from this research project is found in six resources: [30] for computer science research, [14] for NLP, [28] for SA, [29] for Python, [36] for ML, and [3] for NLTK. As the author had no formal training in any of these topics prior to commencing this project, these resources have been invaluable in establishing the base knowledge for completing this project and deserve special recognition. Linguistic and philosophical resources come from research conducted in prior work experience revolving around conversational AI and are consistent with the author's broader views on ontological and epistemological commitments.

5.2 Citations

[1] Anderson, D. R., Sweeney, D. J., Williams, T., Camm, J. D., Cochran, J. J. (2014). Essentials of Statistics for Business and Economics. United States: Cengage Learning.

[2] Audi, R. (2001). The Architecture of Reason: The Structure and Substance of Rationality. Greece: Oxford University Press.

[3] Bird, S., Klein, E., Loper, E. (2009). Natural Language Processing with Python. United States: O'Reilly Media.

[4] Boolos, G. S., Burgess, J. P., Jeffrey, R. C. (2007). Computability and Logic. United Kingdom: Cambridge University Press.

[5] Brandl, Johannes L. and Mark Textor, "Brentano's Theory of Judgement", *The Stanford Encyclopedia of Philosophy* (Winter 2022 Edition), Edward N. Zalta & Uri Nodelman (eds.), URL = <https://plato.stanford.edu/archives/win2022/entries/brentano-judgement/>.

[6] Cambridge Handbook of Cognitive Science, The. (2012). United Kingdom: Cambridge University Press.

[7] Cann, R. (1992). Formal Semantics: An Introduction. United Kingdom: Cambridge University Press.

[8] Chatzyikyriadkidis, S. and Luo, Z., Eds. Modern Perspectives in Type-Theoretical Semantics. (2017). Germany: Springer International Publishing.

[9] Colburn, T. R. (2000). Philosophy and computer science. United Kingdom: M.E. Sharpe.

[10] Cummins, R. (1989). Meaning and mental representation. Cambridge: MIT Press.

[11] Dennett, D. C. (1989). The Intentional Stance. United Kingdom: Penguin Random House LLC.

[12] Dreyfus, H. L. (1972). What computers can't do; a critique of artificial reason. United Kingdom: Harper & Row.

[13] Dybjer, Peter and Erik Palmgren, "Intuitionistic Type Theory", *The Stanford Encyclopedia of Philosophy* (Winter 2024 Edition), Edward N. Zalta & Uri Nodelman (eds.), forthcoming URL = <https://plato.stanford.edu/archives/win2024/entries/type-theory-intuitionistic/>.

[14] Eisenstein, J. (2019). Introduction to Natural Language Processing. United Kingdom: MIT Press.

[15] Hussein, D. M. E.-D. M. (2018). A survey on sentiment analysis challenges. *Journal of King Saud University - Engineering Sciences, 30*(4), 330-338. https://doi.org/10.1016/j.jksues.2016.04.002

[16] Farmer, W. M. (2023). Simple Type Theory: A Practical Logic for Expressing and Reasoning About Mathematical Ideas. Germany: Springer International Publishing.

[17] Goertzel, B. and Pennachin, C., Eds. (2007). Artificial General Intelligence. Germany: Physica-Verlag.

[18] Gopal, M. (2019). Applied Machine Learning. United States: McGraw Hill LLC.

[19] Harris, M. D. (1985). Introduction to Natural Language Processing. United States: Reston Publishing Company.

[20] Haugeland, J. (1985). Artificial Intelligence: The Very Idea. United Kingdom: Penguin Random House LLC.

[21] Heil, J. (2004). Philosophy of Mind: A Contemporary Introduction. Ukraine: Taylor & Francis.

[22] Hill, Winfred F. (1985). Learning: A Survey of Psychological Interpretations. New York: Harper & Row.

[23] Jackendoff, R. (2002). Foundations of Language: Brain, Meaning, Grammar, Evolution. United Kingdom: Oxford University Press.

[24] Janssen, Theo M. V. and Thomas Ede Zimmermann, "Montague Semantics", *The Stanford Encyclopedia of Philosophy* (Summer 2021 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/sum2021/entries/montague-semantics/>.

[25] Knobe, Joshua and Shaun Nichols, "Experimental Philosophy", *The Stanford Encyclopedia of Philosophy* (Winter 2017 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/win2017/entries/experimental-philosophy/>.

[26] Kruse, R., Borgelt, C., Braune, C., Mostaghim, S., Steinbrecher, M., Klawonn, F., Moewes, C. (2016). Computational Intelligence: A Methodological Introduction. United Kingdom: Springer London.

[27] Larson, E. J. (2021). The Myth of Artificial Intelligence: Why Computers Can't Think the Way We Do. United States: Harvard University Press.

[28] Liu, B. (2015). Sentiment Analysis: Mining Opinions, Sentiments, and Emotions. United States: Cambridge University Press.

[29] Lutz, M. (2013). Learning Python: Powerful Object-Oriented Programming. United States: O'Reilly Media.

[30] Mbanaso, U. M., Abrahams, L., Okafor, K. C. (2023). Research Techniques for Computer Science, Information Systems and Cybersecurity. Germany: Springer Nature Switzerland.

[31] McCartney, S. (2001). ENIAC: The Triumphs and Tragedies of the World's First Computer. United States: Berkley Books.

[32] McCawley, J. D. (1988). The Syntactic Phenomena of English, Volume 1. United Kingdom: University of Chicago Press.

[33] McClelland, J. L., Rumelhart, D. E. (1987). Parallel Distributed Processing, Volume 2: Explorations in the Microstructure of Cognition: Psychological and Biological Models. United Kingdom: Penguin Random House LLC.

[34] McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. United States: O'Reilly Media.

[35] Moltmann, Friederike, "Natural Language Ontology", *The Stanford Encyclopedia of Philosophy* (Winter 2022 Edition), Edward N. Zalta & Uri Nodelman (eds.), URL = <https://plato.stanford.edu/archives/win2022/entries/natural-language-ontology/>.

[36] Müller, A. C., Guido, S. (2016). Introduction to Machine Learning with Python: A Guide for Data Scientists. United States: O'Reilly Media.

[37] Newell, Allen, Simon, Herbert, "Computer Science as Empirical Enquiry: Symbols and Search". The Philosophy of Artificial Intelligence. (1990). United Kingdom: Oxford University Press.

[38] Nilsson, N. J. (2009). The Quest for Artificial Intelligence. United States: Cambridge University Press.

[39] Nouwen, Rick, Adrian Brasoveanu, Jan van Eijck, and Albert Visser, "Dynamic Semantics", *The Stanford Encyclopedia of Philosophy* (Fall 2022 Edition), Edward N. Zalta & Uri Nodelman (eds.), URL = <https://plato.stanford.edu/archives/fall2022/entries/dynamic-semantics/>.

[40] Pierce, B. C. (2002). Types and Programming Languages. Ukraine: MIT Press.

[41] Prabowo, R., & Thelwall, M. (2009). Sentiment analysis: A combined approach. *Journal of Informetrics, 3*(2), 143-157. [https://doi.org/10.1016/j.joi.2009.01.003].

[42] Ranta, A. (1994). Type-theoretical Grammar. United Kingdom: Clarendon Press.

[43] Ranta, A. (2011). Grammatical Framework: Programming with Multilingual Grammars. United States: CSLI Publications, Center for the Study of Language and Information.

[44] Russell, S., Norvig, P. (2016). Artificial Intelligence: A Modern Approach. (n.p.): CreateSpace Independent Publishing Platform.

[45] Scarantino, Andrea and Ronald de Sousa, "Emotion", *The Stanford Encyclopedia of Philosophy* (Summer 2021 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/sum2021/entries/emotion/>.

[46] Schank, R. C., Abelson, R. P. (1977). Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures. United States: Lawrence Erlbaum.

[47] Scholz, Barbara C., Francis Jeffry Pelletier, Geoffrey K. Pullum, and Ryan Nefdt, "Philosophy of Linguistics", The Stanford Encyclopedia of Philosophy (Spring 2024 Edition), Edward N. Zalta & Uri Nodelman (eds.), URL = <https://plato.stanford.edu/archives/spr2024/entries/linguistics/>.

[48] Searle, J. R. (1969). Speech Acts: An Essay in the Philosophy of Language. United Kingdom: Cambridge University Press.

[49] Searle, J. R. (1983). Intentionality: An Essay in the Philosophy of Mind. United Kingdom: Cambridge University Press.

[50] Shea, N. (2018). Representation in Cognitive Science. United Kingdom: OUP Oxford.

[51] Sperber, D., Wilson, D. (1995). Relevance: Communication and Cognition. Kiribati: Wiley.

[52] Szabó, Z. G., Thomason, R. H. (2018). Philosophy of Language. India: Cambridge University Press.

[53] Taboada, M. (2016). Sentiment analysis: An overview from linguistics. *Annual Review of Linguistics, 2*(1), 325-347. [https://doi.org/10.1146/annurev-linguistics-011415-040518].

[54] Taylor, K. (1998). Truth and Meaning: An Introduction to the Philosophy of Language. United Kingdom: Wiley.

[55] Tennant, Neil, "Logicism and Neologicism", *The Stanford Encyclopedia of Philosophy* (Winter 2023 Edition), Edward N. Zalta & Uri Nodelman (eds.), URL = <https://plato.stanford.edu/archives/win2023/entries/logicism/>.

[56] Theobald, O. (2017). Machine Learning for Absolute Beginners: A Plain English Introduction. South Africa: Scatterplot Press.

[59] Turner, R. (2018). Computational Artifacts: Towards a Philosophy of Computer Science. Germany: Springer Berlin Heidelberg.

[60] Varela, F. J., Rosch, E., Thompson, E. (1992). The Embodied Mind: Cognitive Science and Human Experience. United States: MIT Press.

[61] von Neumann, J. (2000). The computer and the brain. United Kingdom: Yale University Press.

[62] Wankhade, M., Rao, A. C. S., & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review, 55*(1), 5731-5780. [https://doi.org/10.1007/s10462-022-10144-1].

[63] Wilson, R. (2022). Age of Invisible Machines: A Practical Guide to Creating a Hyperautomated Ecosystem of Intelligent Digital Workers. United States: Wiley.

[64] Wejova, Y. (2009). Sentiment analysis: An overview. *University of Iowa, Computer Science Department*, 5. [https://www.academia.edu/download/3243118/CompsYelenaMejova.pdf].

[65] Wittgenstein, L. (2010). Philosophical Investigations. Germany: Wiley.

6 Appendix A: Python Code for NLTK Solution to Kaggle Sentiment Analysis Dataset

The following is a python-source export from the Jupyter notebook developed as a custom solution for the Kaggle Sentiment Analysis Dataset found at [https://www.kaggle.com/datasets/abhi8923shriv/sentiment-analysis-dataset]. A survey of the ten most popular solutions on the site at the time of this project inspired this solution. Central to the strategy was the use of the NLTK toolkit for formal methods. The intent was not to create an optimal solution, but to explore the basic method for writing code in python and using ML libraries.[29]

```
"""
visona-sentiment-analysis-prototype.py
Jonathan Visona
CPSC 8985-02 - FA2024
python 3.12
"""

# Utility Imports
from string import punctuation as STR_punctuation
import re as REGEX
import warnings as WARN
import matplotlib.pyplot as MPLTLIB_pyplt
import pandas as PNDS
from nltk.tokenize import word_tokenize as NLTK_word_tokenize
from nltk.stem import LancasterStemmer as NLTK_lancaster_stemer
from nltk.corpus import stopwords as NLTK_stopwords
from nltk.stem.wordnet import WordNetLemmatizer as NLTK_wordnet_lemmatizer
from nltk.probability import FreqDist as NLTK_freq_dist
from sklearn.metrics import accuracy_score as SKLN_accuracy_score
from sklearn.metrics import classification_report as SKLN_classification_report
from sklearn.metrics import ConfusionMatrixDisplay as SKLN_confusion_matrix_display
from sklearn.model_selection import train_test_split as SKLN_train_test_split
from sklearn.linear_model import LogisticRegression as SKLN_logistic_regression
from sklearn.ensemble import RandomForestClassifier as SKLN_random_forest_classifier
from sklearn.feature_extraction.text import TfidfVectorizer as SKLN_tfidf_vectorizer
from sklearn.tree import DecisionTreeClassifier as SKLN_decision_tree_classifier

# Global Data
WARN_OFF = False # Turn off after finished debugging
if( WARN_OFF ):
    WARN.filterwarnings( 'ignore' )
ENCODING = 'latin1'
REL_PATH = '../data/'
TRAIN_FILE = 'train.csv'
TEST_FILE = 'test.csv'
MAX_COLS = 10
MAX_ROWS = 3_000_000
HEAD_SIZE = 25
PNDS.set_option( 'display.max_columns', MAX_COLS )
PNDS.set_option( 'display.max_rows', MAX_ROWS )

# Training and Testing Data
training_data = PNDS.read_csv( REL_PATH + TRAIN_FILE, encoding=ENCODING );
test_data = PNDS.read_csv( REL_PATH + TEST_FILE, encoding=ENCODING );
dframe = PNDS.concat( [ training_data, test_data ] )
dframe.head( n=HEAD_SIZE )
dframe.info( verbose=True )
dframe = dframe.drop( columns=[ 'textID','Time of Tweet', 'Age of User', 'Country', 'Population -2020', 'Land
Area (Km²)', 'Density (P/Km²)' ] )
nan_count_per_column = dframe.isna().sum()
print( nan_count_per_column )
dframe.dropna( inplace=True )
nan_count_per_column = dframe.isna().sum()
print(nan_count_per_column)
column_list = dframe.columns.tolist()
print( column_list ); print()
dframe.info( verbose=True )

# Data Conditioning
def alphanumericize( txt: str ) -> str:
    txt = str( txt )
    try:
```

[29] The author had no python or ML experience prior to this research so part of the challenge was acquiring the basics of the constructs of the former and techniques and libraries of the latter.

```
        pattern_replacement_pairs = {
            r'<.*?>': '', # SGML-style tags
            r'https?://\S+|www\.\S+': '', # URLs with http/https
            r'\n': '', # end of lines
            r'[%s]' % REGEX.escape( STR_punctuation ): '',
            r'\s+': ' ', # reduce white-space
            r'[^a-zA-Z0-9\s]': '', # non alphanumerics
            r'\w*\d\w*': '' # lone digits
        }
        for pattern, replacement in pattern_replacement_pairs.items():
            txt = REGEX.sub( pattern, replacement, txt.lower().strip() )
            print( f"{type(txt)=} {txt=}" )
        return txt
    except Exception as e:
        print( f"Error alphanumericizing: {e}" )
        return ""
    return
dframe[ 'alphanumeric' ] = dframe[ 'text' ].apply( alphanumericize )


# Tokenization
def tokenize( txt:str ) -> list:
    try:
        return NLTK_word_tokenize( str( txt ) )
    except Exception as e:
        print( f"Error tokenizing: {e}" )
        return ""
dframe[ 'tokens' ] = dframe[ 'alphanumeric' ].apply( tokenize )
print( dframe[ 'tokens' ] )

# Stopword Removal
def remove_stopwords( txt: str ) -> str:
    txt = str( txt )
    try:
        words = txt.split()
        filtered_words = [ word for word in words if word.lower() not in NLTK_stopwords.words( 'english' ) ]
        filtered_text = ' '.join( filtered_words )
    except Exception as e:
        print( f"Error removing stopwords: {e}" )
        return ""
    return filtered_text
dframe[ 'no_stopwords' ] = dframe[ 'tokens' ].apply( remove_stopwords )

# Lancaster Stemming
stuff_to_be_removed = list( NLTK_stopwords.words( 'english' ) ) + list( STR_punctuation )
stemmer = NLTK_lancaster_stemer()
dframe[ 'lancaster' ] = dframe[ 'no_stopwords' ].tolist()
print( dframe[ 'lancaster' ] )

# Data Conditioning Evaluation
dframe.memory_usage()
dframe[ 'sentiment' ].value_counts()

# Examine Data
dframe[ 'sentiment' ].value_counts().plot( kind='bar' )
dframe[ 'sentiment' ].value_counts().plot( kind='pie' );
word_frq = NLTK_freq_dist( NLTK_word_tokenize( ' '.join(dframe[ 'sentiment' ] ) ) )
MPLTLIB_pyplt.figure( figsize=( 10, 6 ) )
word_frq.plot( 20, cumulative=False )
MPLTLIB_pyplt.title( 'Word Frequency Distribution' )
MPLTLIB_pyplt.xlabel( 'Word')
MPLTLIB_pyplt.ylabel( 'Frequency' )
MPLTLIB_pyplt.show()

# Train and Test Split Model
X = dframe[ 'lancaster' ]
Y = dframe[ 'sentiment' ]
X_train, X_test, Y_train, Y_test = SKLN_train_test_split( X, Y, test_size=0.2, random_state=42 )
X2 = dframe[ 'selected_text' ]
Y2 = dframe[ 'sentiment' ]
X2_train, X2_test, Y2_train, Y2_test = SKLN_train_test_split( X2, Y2, test_size=0.2, random_state=42 )
vectorizer = SKLN_tfidf_vectorizer()
X_val_train = vectorizer.fit_transform( X_train )
X_val_test = vectorizer.transform( X_test )
vectorizer = SKLN_tfidf_vectorizer()
X2_val_train = vectorizer.fit_transform( X2_train )
X2_val_test = vectorizer.transform( X2_test )
```

```
score_baseline = dframe[ 'sentiment' ].value_counts( normalize=True ).max()
print( f"{score_baseline=}" )

# Logistic Regression Model
log_reg = SKLN_logistic_regression( n_jobs=-1 )
log_reg.fit( X_val_train, Y_train )
pred_log_reg = log_reg.predict( X_val_test )
score_log_reg = SKLN_accuracy_score( Y_test, pred_log_reg )
log_reg2 = SKLN_logistic_regression( n_jobs=-1 )
log_reg2.fit( X2_val_train, Y2_train )
pred_log_reg2 = log_reg2.predict( X2_val_test )
score_log_reg2 = SKLN_accuracy_score( Y2_test, pred_log_reg2 )
print( f"{score_log_reg=} {score_log_reg2=}" )

# Classification Report
print( 'LANCASTER' )
print( SKLN_classification_report( Y_test, pred_log_reg ) )
print( 'SELECTED_TEXT' )
print( SKLN_classification_report( Y2_test, pred_log_reg2 ) )

# Confusion Matrix Display
print( 'LANCASTER' )
SKLN_confusion_matrix_display.from_predictions( Y_test, pred_log_reg )
print( 'SELECTED_TEXT' )
SKLN_confusion_matrix_display.from_predictions( Y2_test, pred_log_reg2 )

# Decision Tree Classifier Model
decsn_tree = SKLN_decision_tree_classifier()
decsn_tree.fit( X_val_train, Y_train )
pred_decsn_tree = decsn_tree.predict( X_val_test )
score_decsn_tree = decsn_tree.score( X_val_test, Y_test )
decsn_tree2 = SKLN_decision_tree_classifier()
decsn_tree2.fit( X2_val_train, Y2_train )
pred_decsn_tree2 = decsn_tree2.predict( X2_val_test )
score_decsn_tree2 = decsn_tree2.score( X2_val_test, Y2_test )
print( 'LANCASTER' )
print( f"{score_decsn_tree=}" )
print( 'SELECTED_TEST' )
print( f"{score_decsn_tree2=}" )

# Classification Report
print( 'LANCASTER' )
print( SKLN_classification_report( Y_test, pred_decsn_tree ) )
print( 'SELECTED_TEXT' )
print( SKLN_classification_report( Y2_test, pred_decsn_tree2 ) )

# Confusion Matrix Display
print( Lancaster )
SKLN_confusion_matrix_display.from_predictions( Y_test, pred_decsn_tree )
print( "SELECTED_TEXT" )
SKLN_confusion_matrix_display.from_predictions( Y_test, pred_decsn_tree2 )

# Random Forest Classifier Model
rnd_forst_class = SKLN_random_forest_classifier( random_state=0 )
rnd_forst_class.fit( X_val_train, Y_train )
pred_rnd_forst_class = rnd_forst_class.predict( X_val_test )
score_rnd_forst_class = rnd_forst_class.score( X_val_test, Y_test )
rnd_forst_class2 = SKLN_random_forest_classifier( random_state=0 )
rnd_forst_class2.fit( X2_val_train, Y2_train )
pred_rnd_forst_class2 = rnd_forst_class2.predict( X2_val_test )
score_rnd_forst_class2 = rnd_forst_class2.score( X2_val_test, Y2_test )
print( 'LANCASTER' )
print( f"{score_rnd_forst_class}" )
print( 'SELECTED_TEXT' )
print( f"{score_rnd_forst_class2}" )

print( 'LANCASTER' )
print( SKLN_classification_report( Y_test, pred_rnd_forst_class ) )
print( 'SELECTED_TEXT' )
print( SKLN_classification_report( Y2_test, pred_rnd_forst_class2 ) )
print( 'LANCASTER' )
SKLN_confusion_matrix_display.from_predictions( Y_test, pred_rnd_forst_class )
print( 'SELECTED_TEXT' )
SKLN_confusion_matrix_display.from_predictions( Y2_test, pred_rnd_forst_class2 )

# Cross-Model Evaluation
print( 'LANCASTER EVALUATION\n',
        f'Baseline model = {score_baseline}\n',
        f'Logistic regression = {score_log_reg}\n',
        f'Decision Tree Classification = {score_decsn_tree}\n',
        f'Random Forest Classifier = {score_rnd_forst_class}\n' )
print( 'SELECTED_TEXT EVALUATION\n',
        f'Baseline model = {score_baseline}\n',
        f'Logistic regression = {score_log_reg2}\n',
        f'Decision Tree Classification = {score_decsn_tree2}\n',
        f'Random Forest Classifier = {score_rnd_forst_class2}' )

# Test and Validation Set on Models
def to_lower( txt: str ) -> str:
    return txt.lower()
def test_items( items: list ):
    dframe = PNDS.DataFrame( {'test': items } )
    dframe[ 'lower' ] = dframe[ 'test' ].apply(to_lower)
    new_x_val_test = vectorizer.transform( dframe[ 'lower' ] )
    return new_x_val_test
print(test_items(["I am happy", "I am here", "I am sad"]))
```