Towards Evaluating Creativity in Language

Matey Krastev

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Bachelor of Science
of the
University of Aberdeen.

Department of Computing Science

2022

Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Signed:

Date: 2022

Abstract

An expansion of the title and contraction of the thesis.

Acknowledgements

Much stuff borrowed from elsewhere.

Contents

1 Introduction **7**

1.1 Inspiration . . . 7
1.2 Motivation . . . 7
1.3 Objectives . . . 8
1.4 Contributions . . . 8

2 Related Work **9**

2.1 Challenge Landscape . . . 9
2.1.1 Part of Speech Tagging . . . 10
2.1.2 Word Sense Disambiguation . . . 12
2.1.3 Sentiment Analysis . . . 12
2.1.4 Named Entity Recognition . . . 12
2.1.5 Phonetic Analysis . . . 12
2.1.6 Natural Language Generation . . . 12
2.2 Creative Measures . . . 12
2.3 Tools . . . 13

3 Methodology **14**

3.1 Datasets . . . 14
3.1.1 Brown Corpus . . . 14
3.1.2 Project Gutenberg . . . 15
3.1.3 Hierarchical Neural Story Generation . . . 15
3.1.4 Discarded Datasets and Corpora . . . 15
3.2 WordNet . . . 16
3.2.1 Numerical representations of semantic tokens . . . 16
3.3 Metrics . . . 17
3.3.1 Number of Words . . . 17
3.3.2 Number of Sentences . . . 18
3.3.3 Word Length . . . 18
3.3.4 Sentence Length . . . 18
3.3.5 Number of Tokens . . . 19
3.3.6 Concreteness . . . 19
3.3.7 Imageability . . . 19
3.3.8 Frequent Word Usage . . . 20

3.3.9 Proportion of Parts of Speech 21

4 Design **22**
4.1 Requirements 22
4.1.1 Scenarios 22
4.1.2 Functional Requirements 22
4.1.3 Non-functional Requirements 23
4.1.4 Memory Requirements 23
4.2 Technology Choices 23
4.2.1 Python 23
4.2.2 PyTorch 24
4.2.3 NLP Frameworks 24
4.3 Code Style 24
4.3.1 PEP8 24
4.3.2 Docstrings 25
4.3.3 Linting 25
4.3.4 Testing 25
4.3.5 Code Review 25
4.3.6 Version Control 25
4.4 Documentation 25
4.4.1 Documentation Framework 25
4.4.2 Sphinx 25
4.4.3 Hosting 25

5 Implementation **26**

6 Evaluation **27**
6.1 Experimental Design 27
6.1.1 Datasets 27
6.2 Experiments 28
6.2.1 Document Class Identification 28
6.2.2 Authorship Identification 29
6.2.3 Machine-Generated Text Identification 31

7 Discussion **34**
7.1 Formal evaluation of the system 34
7.1.1 Accomplished goals 34
7.1.2 Limitations of the system 34

Chapter 1

Introduction

1.1 Inspiration

At the heart of the field of artificial intelligence is the concept of reproducing aspects defining human intelligence through rigorous examination and replication of the mechanisms that drive progress. This is a uniquely multi-faceted problem with a multitude of approaches, each tailored to a very specific aspect or manifestations of intelligence. Naturally, intelligence assumes following a logical pathway to arrive at sensible conclusions that interact with the real world beneficially. This can be viewed through the lens of methodical, defined process that always follows a certain formula. An organism, assumed to be intelligent, might always follow such formulaic actions given a set of prerequisite conditions to accomplish a defined goal. Certain schools of thought theorise that there is such an order in every little action, and such thread of logicality interweaves every law of nature, known or not. Then, follows the question, can we recognize and define such a thread for the ambitious field of creativity? We seek not to properly define or constrain the subject of creativity, rather, we explore markers of what could only be a subset of the very broad field of human creativity.

The subject of the present document is exclusively the study of linguistic creativity. Henceforth, we seek to: confirm prior results of the research of psycholinguistics, affirm that hypotheses and conclusions drawn from them correlate highly with certain manifestations or aspects of creativity, and make firm the subject of creativity, that is, provide tools that may be used for exploration and analysis of specific creative features found in text.

1.2 Motivation

Large language models (LLMs) are probabilistic models of language widely used for most tasks in the field of Natural Language Processing (NLP), ranging from machine translation, text classification, sentiment analysis, auto-completion, error correction, or even simple dialogue communication. However, because, fundamentally, LLMs operate on probabilities, a lot of the applications utilizing them tend to struggle with generating logically coherent novel sequences.

Large language models are usually trained on enormous language corpora, mined from books and articles (Gerlach and Font-Clos, 2018), social media posts (Derczynski et al., 2016), or otherwise internet crawls (Gao et al., 2021). Therefore, the underlying assumption would be that, in their attempt to replicate language, they could find some success at least in terms of generating basic structure in creative fields of work such as writing code (Chen et al., 2021) or screenplays (Mirowski et al., 2022), among others.

While LLMs display convincingly human-like text in such areas, we instead intend to examine their ability to produce specifically *creative* text. By its very nature, creativity tends to be a subjective matter, thus, we aim to identify specific aspects of creativity that can be more commonly found within creative text, such as poetry or short stories, as opposed to other less creatively-oriented genres, e.g. news articles or user manuals. Having identified specific linguistic markers within human-written creative texts, we can then begin to evaluate the extent to which LLMs exhibit these properties in generated text. This in turn has the potential to inform techniques to adapt existing text-generation models for applications involving creative language. Therefore, we seek to compile and produce a software package that can efficiently and accurately represent and evaluate linguistic creativity.

1.3 Objectives

We set out to investigate specific markers defining or correlating with conventional creativity in language. As some aspects of creativity have been investigated by research disciplines such as the field of psycholinguistics, we seek to filter and compile a set of measures that have been shown to correlate more highly with creative texts in the literature. Following that, we aim to evaluate the extent to which current LLMs can provide insights into capturing creative elements or patterns within writing. Finally, if time allows, we may use these findings to explore how natural language generation can be adapted to produce texts exhibiting more human-like levels of the creative attributes we have discovered. These can be summarised by the following three questions:

- Can psycho-linguistically motivated measures (that is, the explored metrics) successfully characterise creative properties in language?
- Can a machine learning approach be adapted for evaluating creativity in natural language?
- Can we influence a subset of current LLMs to exhibit more creative traits as defined by our creativity metric via some conventional approaches such as hyperparameter tuning or varying decoding strategies?

For the first two, we develop a suite of benchmarks we shall distribute and report the results of. For the third question, we plan to train a model maximising performance on the developed benchmarks. We will then report our findings on the best-performing models and share the architecture.

1.4 Contributions

The contributions of this work are as follows:

[MK: <insert rolling tumbleweed>]

[MK: Insert explanation of the sections]

Chapter 2

Related Work

The field of creativity research has been studied for decades, and there are many different approaches to the problem. In this section, we will discuss the different approaches to creativity research, and how they relate to our work. We will also discuss the different approaches to creativity in the context of natural language processing, and how they relate to our work.

2.1 Challenge Landscape

Most of the work going on in our minds as we read a given text is unconscious. We automatically parse characters and tokenize the text into words, sentences, paragraphs, and so on. At the same time, we apply tagging to understand actors (subjects and objects), actions (verbs), setting and situation (adverb), and also try to understand the sense a given word is used in, and then we transform the connotations or the sense of words inside our own little mind representation of the words. We also apply sentiment analysis to understand the emotional state of the text (or speech), and we apply phonetic analysis to understand the pronunciation of the words, as we read them out in our minds, e.g. when reading a book. Note that we have constrained the example of autonomous processes to reading, although these subconscious processes happen regardless of whether we read, write, or speak. Fact of the matter is, language is inherently a complex social construct that takes years to learn, even more to master, and more than a lifetime to perfect. Because it is so expressive and hard to grasp, many rules have been invented and applied to constrain or clearly define the boundaries of the language (e.g. English), so that, within a given language speaker group (e.g. the space of English speakers), the biggest common denominator of people may understand what is being said by the others. The rules may additionally have their own rules and exceptions to the rules.

Therefore, there are multiple approaches being applied to natural language processing tasks. The more classical one, and the one that has been applied for the larger part of the developments in the field, has been the rule-based method. The approach seeks to use clearly defined rules to parse and understand the linguistic features of the given text. It is a strong approach, as language has been strongly studied for centuries and the aforementioned rules and constraints have already been applied to it. For languages with few changing features or slowly changing features, it performs acceptably well on most NLP tasks, and not far off from human speakers. [expand with 1-2 paragraphs and **references pls**]

Another approach or a subclass of the rule-based approach, is the **statistical method** [MK: this statement is not bs, right?]. The statistical method seeks to find patterns inside language as a whole. For example, the word “wind” may appear both as a noun and a verb – that is, the meaning of the word may be ambiguous – but the noun form is much more prevalent. Therefore, in tasks such as part of speech (PoS) tagging, some algorithms prefer to use the most common type of PoS class a given word occurs as, in a sufficiently large corpus of text. We do come back to the PoS tagging example later on. Alternative example of the statistical method being applied would be translation. Previous approaches to machine translation (MTL) included learning frequency of words and phrases occurring together (and how those map to counterparts in the language being translated to). For example:

> Finding Nemo is like finding fish in a school of fish.

A naive approach to the translation of this sentence would consider school as its most common (noun) definition - that of place of learning for *humans* and translated the word literally. A more sophisticated approach to translation, applying not just simple rules to translating, would consider the whole phrase “school of fish”, and would have understood that it refers to a countable form of the word fish (relating to a large number of fish), and therefore translated the phrase as a whole to the target language.

The most current approach to many of the challenging NLP tasks (text summarization, machine translation, speech recognition, etc.) is a machine-learning based one. The motivation is multifold: firstly, most of the work that goes on in our minds as we read a given text is unconscious and automatic, just as we do not have to consciously intend to breathe in order to breathe. In much the same way, we rarely consciously make an effort to understand every part of the text, and many of the details behind understanding language are vague and ambiguous (consider how someone would respond if they are asked to explain their thought process behind parsing a given text). Secondly, more in line with the topic of our research, we do not have a good (objective) way to measure the quality of the work. It is subjective and difficult to measure. Not to mention, behaviour – and consequently, consciousness – is something that arises from environment, upbringing, culture, and so on. Yet, value for quality is something that multiple individuals can share – many people can have a sense of appreciation for a novel they read or a speech they heard (of course, usually for slightly different reasons and perceptions) – but there tend to be common elements that people widely enjoy seeing and experiencing.

The intuition of machine learning and deep learning is that you can try to replicate the unconscious logical circuitry going on behind the scenes without having a very solid grasp of the exact logic behind it. Therefore, in fields with few or changing rules, such as linguistics, music, and image processing, the machine learning approach tends to find large success, and has even been shown recently to be able to perform very similarly to humans (Bubeck et al., 2023). All in all, we cannot ignore the potential for machine learning to be applied to the field of creativity, and we explore this potential in our work.

We, therefore, go into detail on some of the methods we utilize in the project, and the various approaches that have been taken in the past.

2.1.1 Part of Speech Tagging

As we have established, a word may play a different role depending on its position in the sentence, both absolute and relative to the other words. Words that denote objects or persons, we generally call **nouns**, while words that denote (usually active) actions, we call **verbs**, and so on. Part-of-speech tagging as a task poses the challenge of assigning to the sequence of tokens (we use "tokens" and "words" interchangeably) $x_1, x_2, \ldots, x_n$, a sequence of tags $y_1, y_2, \ldots, y_n$, where each token x_i has a corresponding tag y_i, making tagging a task of disambiguation, that is, removing ambiguities in text.

The phenomenon of ambiguity is hardly exclusive to the English language, although it is a key problem that we would need to take head-on, as many of the tasks we do want to tackle, rely on POS tagging, in order to avoid needless computations (and in consideration of the limited computing resources for both end-users and ourselves, as we may do larger-scale experiments). Human performance on POS-tagging has been shown to be around 97% accuracy (Manning, 2011) for English texts. We will use this as a base of comparison with automatic algorithms.

Types:	**WSJ**	**Brown**
Unambiguous (1 tag)	44,432 (86%)	45,799 (85%)
Ambiguous (2+ tags)	7,025 (14%)	8,050 (15%)
Tokens:		
Unambiguous (1 tag)	577,421 (45%)	384,349 (33%)
Ambiguous (2+ tags)	711,780 (55%)	786,646 (67%)

Table 2.1: Tag ambiguity in the Brown and WSJ corpora (Treebank-3 45-tag tagset). Adapted from Jurafsky and Martin (2009)

.

As seen in Table 2.1, many of the words in the tag-annotated corpora are, in fact, unambiguous. That is, the given word appears only as one single *type* of part of speech. What we may miss, however, is the fact that the actual *tokens*, the words, are mostly ambiguous (55% on WSJ and 67% on Brown). For example, *elephant* can only appear as a noun, but the word *back* can appear as either a verb, an adjective, a noun, or an adverb. And *back* is a much more common word than, say, *elephant*. Given this example, we can then proceed to potential solutions.

Generally, the issue is hardly new in the field of text mining, and solutions have been attempted since long ago in the past. The Brown corpus by Francis and Kucera (1979) is a manually annotated for parts of speech corpus of American English texts from variety of genres, and has been widely used as a benchmark for algorithms for POS-tagging – at least ones focusing on the English language.

One baseline metric is the **most frequent class** classifier. This method assigns to each word the most frequent tag it appears as in some training corpus. The metric has been shown to be 92% effective (Jurafsky and Martin, 2009) in correctly classifying the tags of a given text, which is just 5% below human accuracy, as described above. However, we can do better.

Hidden Markov Models are among the more successful ones, as demonstrated by Schütze and Singer (1994); Thede and Harper (1999); Nigam et al. (1999). The idea stems from the fact that the probability of a word x_i appearing in a given position in a sentence, is dependent on the word that came before it. This is a Markov assumption, as it assumes that the probability of a word appearing in a given position is independent of the words that come after it. The given assumption is a strong one, and it is not always true. However, it is a good starting point, and it has been shown to be effective in many cases. The

While the HMM is a useful and powerful model, it turns out that HMMs need a number of augmentations to achieve high accuracy, as exemplified by authors such as Goldberg et al. (2008). Brants (2000) demonstrate between 96 and 99% accuracy for HMMs, while Thede and Harper (1999) demonstrate an accuracy between 96-98%, and between 88 and 93% accuracy on various datasets by Goldberg et al. (2008). For example, in POS tagging as in other tasks, we often run into unknown words: proper names and acronyms are created very often, and even new common nouns and verbs enter the language at a surprising rate – the Oxford English Dictionary has added 700 new words to its definitions just in its most recent quarter-yearly update in March (Oxford English Dictionary Editorial, 2023).

Conditional Random Fields (CRFs) have been proposed to deal with the following motivations: the adding of arbitrary features, e.g. based on capitalization or morphology (words starting with capital letters are likely to be proper nouns, words ending with -ed tend to be past tense (VBD or VBN), etc.); knowing the previous or following words (if the previous word is "the", the current one is not likely to be a verb).

State-of-the-art POS taggers use neural networks behind the hood, being either bidirectional RNNs or Transformers like BERT.

2.1.2 Word Sense Disambiguation

[both of these can be included as potential metrics]

2.1.3 Sentiment Analysis

2.1.4 Named Entity Recognition

2.1.5 Phonetic Analysis

2.1.6 Natural Language Generation

- top KP sampling
- challenges

2.2 Creative Measures

The field of creativity has been broadly studied, initially by psychologists, and more recently by computer scientists. Intuitively, the nature of creativity is a subjective matter, and therefore, it is difficult to define. However, there are some commonalities that can be found in creative works. For example, creativity is often associated with novelty, and is often associated with the ability to generate new ideas. Franceschelli and Musolesi (2022), for example, determine the three factors of creativity as value, novelty, and surprise, and then explore machine learning approaches to measuring creativity. We agree with them, but decide not to limit ourselves to just these three. However, their contributions provide insights we can use in our work. [MK: did we use it - if we did not, delete this sentence]

- Burstiness of verbs and derived nouns: Patterns of language are sometimes 'bursty' Pierrehumbert (2012). This paper presents an analysis of text patterns for domain X. measures include XYZ...
-

2.3 Tools

Chapter 3

Methodology

In this chapter, we explore the different datasets to be used, the methods for evaluating creativity, and the algorithms for creativity evaluation. We will furthermore discuss the strengths and limitations of the proposed methods and algorithms, including time complexity, memory constraints, and the accuracy of the results. We take an informed approach to the selection of the datasets and the methods for evaluating creativity, and discuss the reasons for our choices in detail, and support them with relevant literature as explored by other researchers in the field.

3.1 Datasets

Datasets are vital for the success of any given project in the field of machine learning, and even more so when concerning linguistics. As evidenced by Torralba and Efros (2011), the quality of the data used for training a model has a direct impact on the quality of the results. A model trained on a specific dataset, e.g. a corpus of law documents, can be expected to perform poorly on a dataset of medical documents, as the two domains are inherently different. Thus, we take particular care in planning and selecting the datasets we use. We also consider ease of use and access, as some datasets may require additional processing, others are subject to availability issues (e.g. paid datasets and corpora), and some may be too large to be used in a reasonable amount of time. In this section, we will explore the datasets used in this project, and discuss their strengths and limitations.

3.1.1 Brown Corpus

The Brown Corpus (Francis and Kucera, 1979) is a widely used corpus in the field of computational linguistics, noted for the small variety of genres of literature it contains. The Corpus itself is founded on a compilation of American English literature from the year 1961. It is also small in terms of size, totalling around one million words, at least compared to modern corpora, which we also explore later on. The corpus also suffers from the issue of recency, as the works and language may be outdated for modern speakers of English.

Of interest is the fact that the corpus has been manually tagged for parts of speech, a process that tends to be error-prone. As we will see later on, this fact has implications in terms of the supervised learning algorithms we implement for creativity evaluation. Still, we opt to utilize it primarily for prototyping purposes and drawing preliminary conclusions about the effectiveness of the implemented algorithms, rather than in-depth analysis and publication of results.

3.1.2 Project Gutenberg

Project Gutenberg[1] is a large collection of more than 50,000 works available in the public domain. The collection contains literature from various years and various genres and thus is suitable for training and evaluation of the developed benchmarks in the context of creativity study.

As the Project does not offer an easy to process copy of its collection, we turn to the work of Gerlach and Font-Clos (2018). The team developed a catalogue for on-demand download of the entire set of books available on the Project Gutenberg website, intended for use in the study of computational linguistics. The tool avoids the overhead of writing a web-scraper or a manual parser for the downloadable collections of Project Gutenberg books made available by third parties, as well as enables easy synchronization of newly released literature. Instead, we are only required to develop a simple pipeline for the data to be fed into the utilized systems.

3.1.3 Hierarchical Neural Story Generation

In their work, Fan et al. (2018) trained a language model for text generation tasks on a dataset comprised of short stories submitted by multiple users given a particular premise (a prompt or a theme) by another user. [MK: Give an example for how one such short story would look like.] The dataset in question is technically referred to a series of posts and comments (threads) to them on the popular social media platform REDDIT, and more tightly, the *subreddit* forum R/WRITINGPROMPTS. The authors of the work Fan et al. (2018) have made the dataset available for public use, and we have used it for the purpose of evaluating the performance of our creativity benchmarks. As described by the authors on their GitHub page[1], the paper models the first 1000 tokens (words) of each story.

[MK: How do we use this dataset? You should describe the process of how we use it.]

3.1.4 Discarded Datasets and Corpora

Some datasets were considered, however, discarded due to: not being deemed applicable for the context of the application; general lack of availability of the dataset in a form that is easily accessible for our purposes; simply being infeasible to use due to the size of the dataset and the hardware constraints imposed on the project; or other reasons of similar nature.

The COCA

The Corpus of Contemporary American English (COCA)[2] is a large corpus of American English, containing nearly 1 billion words of text from contemporary sources. It is a collection of texts from a variety of genres, including fiction, non-fiction, and academic writing. The corpus offers a variety of tools for analysis of the data, including a concordance tool, a word frequency list, and a collocation finder. Naturally, many of those tools could be used in the field of statistical creativity analysis that we explore.

The corpus does offer limited access to the full API, as well as free samples of the data, however, the full corpus is not available for free, and the cost of acquiring it is prohibitive for the limitations set forward by the project. Nevertheless, the corpus is a valuable resource for the field of computational linguistics, and we would like to explore it further given less constraints.

[1] https://www.gutenberg.org/
[1] https://github.com/facebookresearch/fairseq/blob/main/examples/stories/README.md
[2] https://www.english-corpora.org/coca/

3.2 WordNet

WordNet(Fellbaum, 1998) is a lexical database of semantic relations between words that links words into semantic relations including synonyms, hyponyms, and meronyms. The synonyms are grouped into synsets (sets of synonyms) with short definitions and usage examples. It can thus be seen as a combination and extension of a dictionary and thesaurus (Wikipedia contributors, 2023b).

For our specific use cases, we have identified it as a valuable resource in terms of relational representation of words in semantic space. In the given context, this enables us to traverse a semantic graph for synonyms and related words for the goal of enriching potential similarity between the set of creative parts of speech (i.e., nouns, adjectives, adverbs), which we narrow down our scope to in particular.

3.2.1 Numerical representations of semantic tokens

[MK: Potentially move this section to background work] The idea of representing words or lexical tokens as numerical vectors (or even scalars) is hardly new. For example the SimLex-999 dataset (Hill et al., 2015) gives values on a scale from 0 to 10, like the examples below, which range from near-synonyms (vanish, disappear) to pairs that scarcely seem to have anything in common (hole, agreement):

word1	word2	score
vanish	disappear	9.8
hole	agreement	1.2

Table 3.1: Example Simlex-999 pairs

Early work on affective meaning by Osgood et al. (1957) found that words varied along three important dimensions of affective meaning:

- valence: the pleasantness of the stimulus
- arousal: the intensity of emotion provoked by the stimulus
- dominance: the degree of control exerted by the stimulus

Osgood et al. (1957) noticed that in using these 3 numbers to represent the meaning of a word, the model was representing each word as a point in a three-dimensional space, a vector whose three dimensions corresponded to the word's rating on the three scales. This revolutionary idea that word meaning could be represented as a point in space (e.g., that part of the meaning of heartbreak can be represented as the point $[2.45, 5.65, 3.58]$) was the first expression of the vector semantics models that we introduce next. [MK: **You can paraphrase this**]

Word2Vec

Mikolov et al. (2013) show in their work that words may be represented as dense vectors in N-dimensional space, and we can perform mathematical operations on them that may yield effective results in terms of word representation.

Measuring distance in vector representations of semantic tokens

Intuition tells us that the dot product of vectors in N-dimensional space will grow when the set of vectors has similar values and decrease when the values are not similar. Thus, we can then construct the following metric for semantic similarity between vector representations of words:

$$D(v,w) = v \times w = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \cdots + v_N w_N$$

The current metric, however, suffers from the problem that vectors of higher dimensions will inevitably be larger than vectors with lower dimensions. Furthermore, embedding vectors for words that occur frequently in text, tend to have high values in more dimensions, that is, they correlate with more words. The proposed solution is to normalize using the **vector length** as defined:

$$|v| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

Therefore, we obtain the following:

$$\text{Similarity}(v,w) = \frac{v \times w}{|v||w|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

This product turns out to be the same as the cosine of the angle between two vectors:

$$\frac{a \times b}{|a||b|} = \cos(\theta)$$

Therefore, we will call this metric the **cosine similarity** of two words. As mentioned, the similarity grows for vectors with similar features along the same dimensions. Note the boundaries of said cosine metric: we get -1 for vectors which are polar opposites, 0 for orthogonal vectors, and 1 for equivalent vectors. Of note is the fact that such learned vector embeddings only have values in the positive ranges, thus, it is impossible to have negative values for the cosine similarity ($\text{Similarity}(a,b) \in [0,1]$).

Contrary to it, we also identify the metric of **cosine distance** between two vectors, as one minus the similarity of the vectors, or:

$$\text{Distance}(v,w) = 1 - \text{Similarity}(v,w)$$

The cosine distance may prove useful when dealing with minimisation problems as is often the case with machine learning.

3.3 Metrics

3.3.1 Number of Words

The total number of words in a given piece of text. At first glance, this metric does not impress and is, in fact, exceedingly simple. But that is fine – we do not always need complex metrics. Sometimes, even a trivial metric as this one can inform a lot about the structure of the text. For example, the number of words in a text is directly correlated with the length of the text. This can be useful in determining the complexity of the text, as well as the time it takes to read it. In some uses, for example, comparing between books and *Twitter* posts, we do not need much more information to recognize that these texts belong to entirely different genres. Such a metric is a good complement to and often used in conjunction with other metrics.

3.3.2 Number of Sentences

The number of sentences, similarly to number of words, is a trivial measure for the length of the text. However, it can be used to determine the complexity of the text. For example, a text with a large number of sentences is likely to be more complex than a text with a small number of sentences. This is because a text with a large number of sentences is likely to involve longer intellectual activity. Of course, in light of recent developments in the field of natural language generation, this metric is not particularly useful. However, due to how trivial to implement it is, it can be used in conjunction with other metrics for text classification tasks.

3.3.3 Word Length

> *"Because even the smallest of words can be the ones to hurt you, or save you."* –
> Natsuki Takaya

Word length fills in the set of trivial metrics we implement for text benchmarking. The intuition is simple. Given a sufficiently large corpus, the average word length – that is, the number of characters in a word – will converge to a certain number – in English, this number tends to be between 4 and 5. Any deviations, either positive or negative, from this norm can then be used to determine the complexity of the text. For example, a text with a large number of long words is likely to be more complex than a text with a large number of short words. Naturally, words expressing more specific concepts tend to have a longer character length than words we use in general speech and are sometimes ambiguous. This phenomenon is established in English, although the essence may not generalize well for other languages, e.g. Chinese and Japanese, where a single character can generalize to a whole word or a concept as a whole, but given that we are working in the context of the English language, we are not concerned with this issue.

3.3.4 Sentence Length

Similar to word length above, the average sentence length is a trivial metric describing the number of characters per sentence. Intuition tells us it will be closely related to the average word length, but also indicative of text features such as complexity and readability. For example, legal documents tend to have longer sentences than, say, newspaper articles. This is because legal documents tend to be more complex and require more time to read and understand. In contrast, newspaper articles tend to be more accessible and are written in a way that is easy to understand.

Writers may also be interested in this metric, as very long sentences are often difficult to read and understand, as the reader may lose track of the subject of the sentence among the many objects, actions and modifiers; not to mention unnecessary punctuation where simply beginning a new sentence would be far more readable... a useful feature like this can pinpoint such writing issues, inform writers where they may cut or simplify their sentences, and in general help them improve their writing style – a feature that is often overlooked in the context of text understanding – this is also the longest sentence in the entire document.

3.3.5 Number of Tokens

Completing the set of trivial metrics is the general number of tokens in the text. The metric correlates highly with average sentence length and word length. Rather than counting characters in the sentence or word length, however, we take a look at the number of tokens encountered in the text, usually at the sentence level.

3.3.6 Concreteness

Concreteness is the degree to which a word refers to a tangible object or a concrete idea. For example, the word *apple* is concrete, while the word *time* is abstract. Brysbaert et al. (2014) provide a dataset of concreteness ratings for 40,000 English lemmas (English words and 2,896 two-word expressions (such as “zebra crossing” and “zoom in”), obtained from over four thousand participants by means of a norming study using internet crowdsourcing for data collection). The dataset is based on the concreteness ratings of the four thousand participants, who rated the concreteness of 40,000 words on a scale from 1 to 5. The concreteness of a word is measured on a scale from 1 to 5, where 1 is the most abstract and 5 is the most concrete:

[MK: direct citation of the study, if i need to paraphrase it, probably would delete it]

> Some words refer to things or actions in reality, which you can experience directly through one of the five senses. We call these words concrete words. Other words refer to meanings that cannot be experienced directly but which we know because the meanings can be defined by other words. These are abstract words. Still other words fall in-between the two extremes, because we can experience them to some extent and in addition we rely on language to understand them. We want you to indicate how concrete the meaning of each word is for you by using a 5-point rating scale going from abstract to concrete.

The dataset provides norms for the 40,000 words and 2,896 two-word expressions – including mean and standard deviation for each entry.

The intuition of this metric is that a word that is more concrete is more likely to be used in a creative context, as it is easier to imagine and relate to. It not only describes one aspect of the word’s meaning, but authors (and genres, in general), tend to exhibit specific characteristics, such as legal documents being more generally more concrete - one would expect concrete objects and entities to appear more in documents such as the UN Human Rights Charter, or protocols for health standards control, for example.

3.3.7 Imageability

Imageability is the degree to which a word evokes a mental image, as described by de Groot (1989). For example, the word *apple* is more imageable than the word *time*. Brysbaert et al. (2014) provide a dataset of imageability ratings for 9,000 English lemmas (English words and 2,896 two-word expressions (such as “zebra crossing” and “zoom in”), obtained from over four thousand participants by means of a norming study using internet crowdsourcing for data collection). The dataset is based on the imageability ratings of the four thousand participants, who rated the imageability of 9,000 words on a scale from 1 to 5. The dataset also contains the number of participants who rated each word, and the standard deviation of the ratings.

3.3.8 Frequent Word Usage

"Separate text from context and all that remains is a con."— Stewart Stafford

Word frequency refers to the number of times a given word appears in a given context. Word frequency naturally differs from text to text, and smart word choice in general is an excellent indicator for intellectual linguistic use. The intuition behind selecting this metric is that words that are occurring less frequently in common speech are more likely to be used in a creative context. To give an example by rewording the last sentence, would yield: "The intuition behind identifying this linguistic measure owes to the words' property of inverse proportionality between frequency and perceived creative or intellectual value."

As noted, less common words are associated with higher perceived intellectual value. Even more so, the use of less common collocations (words occurring very close in a given context) hints at a higher level of linguistic skill. Of course, simply chaining completely unrelated words together (e.g. "palmarian tobaccophile ephemeron urbarial") hints not to high intellectual value, but rather to spitting out a random sequence of words. Properly applied in context, though, commonly not associated words can be used to great effect. This is especially true in the case of poetry, where the use of uncommon words and collocations is a common practice, or, for example, in biological contexts, such as medicine and botany, where very precise yet niche namings and conventions are mandated. This type of dissonance between common speech and niche terminology is a common theme in creative writing, and is often used to great effect. For example:

> *"When they'd gone the old man turned around to watch the sun's slow descent. The Boat of Millions of Years, he thought; the boat of the dying sungod Ra, tacking down the western sky to the source of the dark river that runs through the underworld from west to east, through the twelve hours of the night, at the far eastern end of which the boat will tomorrow reappear, bearing a once again youthful, newly reignited sun."*
>
> – *The Anubis Gates*, Tim Powers

In this context, "boat" is a completely valid and understandable synonym of the word "sun", yet the word "boat" co-occurring with the word "sun" outside this context is not common, and therefore, we are prompted to believe that this context is more 'creative'.

We tackle the topic of contextual surprise further on with subsequent metrics, but for now, we focus on the general idea of individual word frequency.

Given a sufficiently large linguistic corpus, we obtain a list of words and their frequency of occurrence. We can then use this list to calculate the frequency of occurrence of a given word in a given text. We can then use this frequency as a metric for the text's creativity. Choice of corpus is key here, as the corpus should be large enough to contain a wide variety of words, but not specialized enough to inflate the frequency of niche words. For example, a corpus of medical texts would contain a lot of medical terminology, which would inflate the frequency of medical terms, and therefore, would not be a good choice for a general creativity metric, for example in the case of a poetry contest.

For our use case, we opt to use the British National Corpus (BNC) (BNC Consortium, 2007), which is a 100 million word collection of samples of written and spoken language from a wide range of sources, designed to represent a wide cross-section of British English from the later part of the 20th century, both spoken and written. The BNC is a good choice for our use case, as it is a general corpus, and contains a wide variety of words, but is not specialized enough to inflate the frequency of niche words.

The frequency lists we use are provided by the work of Leech et al. (2014) and are readily available in sheet form for both lemmatized and non-lemmatized words. In our case, we attempt to adhere only to the lemmatized versions in order to have consistency with previous metrics, but also to have normalized results, e.g., although the words ‘am’, ‘is’, ‘are’ are all inflections of the verb ‘to be’, they may have different frequencies and different positions in the list. POS tagging and lemmatization again come into play here, as we need to be able to identify the lemma of a given word in order to find its proper frequency in the list. The frequency lists indicate the words’ frequencies per 100 million tokens. Intuitively, given a varied enough corpus such as the BNC, we expect these numbers to normalize and generalize well for general English. We then use the frequencies for the lemmas and the take the logarithm with base 10 of the given frequency like so:

$$\text{freq}(x) = \log_{10}(\text{Frequency}_{BNC\ 1M}(\text{Lemma}(x))) \tag{3.1}$$

Like before, if a word does not appear in the BNC, we discard it and continue. We then calculate the average frequency of the words in the text, and return the metric for interpretation by the end user.

3.3.9 Proportion of Parts of Speech

Chapter 4

Design

In the following chapter, we introduce concepts behind the design of the developed library and how those will be implemented in the application. We also discuss the technology choices we have made and the reasoning behind them. Furthermore, we take a look at how the application will be structured and how it will be distributed, and how the users will be able to interact with it via a command-line interface, or apply declared methods and classes inside their own applications. Finally, we discuss the delivery of a documentation and a user guide, as well as the testing and validation of the application.

4.1 Requirements

In this section, we detail the requirements regarding the application and the library in the following sections. We will also discuss the requirements regarding the documentation and the user guide, as well as the requirements regarding the testing and validation of the application.

4.1.1 Scenarios

We use scenarios to better illustrate the use-cases of the MAD HATTER package. They provide a frame for the functional and non-functional requirements and allow us to nail down the specific requirements and pain points, well in advance of the de facto launch of the package.

1. Evaluating linguistic features associated with creative aspects of language in a given text. Users should be able to clearly present a text and evaluate it against a set of metrics, which will then be presented in a clear and concise manner.

2. Providing methods for interacting, plotting and visualizing the results of the evaluation. Users should be able to easily interact with the results of the evaluation, and plot them in a way that is easy to understand and interpret.

3. Providing a pipeline for batch data analysis to be used in larger-scale linguistic research operations. Users should be able to easily process large amounts of data in a batch manner, and then interact with the results of the evaluation in a way that is easy to understand and interpret.

4.1.2 Functional Requirements

Functional requirements pinpoint the specific tasks that the application needs to be able to perform. They are the most important part of the requirements, as they are the ones that will be used to evaluate the success of the application. We will list the functional requirements below.

Accuracy, etc.

1.

4.1.3 Non-functional Requirements

The package henceforth needs to satisfy a list of viable non-functional requirements, which we will list below.

Efficacy

The implemented algorithms must be viable to deploy both in small-scale and for large-scale applications. This means that the algorithms must be able to scale to large amounts of data, while also being able to run on a single machine. In our case, a user should be able to quickly evaluate their texts across several metrics, however, as we also strive to apply this benchmark to large-scale corpora, we must also ensure that the algorithms are scalable. We will therefore not only seek to reiterate on the existing literature and implement the most promising algorithms for the task of creativity evaluation, but also optimize them for deployment on HPC clusters.

4.1.4 Memory Requirements

The old adage that *memory is cheap* is not entirely true. While it is true that memory is cheap, it is also true that memory is not free (*and no, we cannot "just download more RAM"*). In fact, some LLMs simply tend to be too large to reliably fit within the memory constraints of a personal computer. [MK: We should probably cite some sources here.] Furthermore, model accuracy tends to grow with the size of the neural network and the size of the used vocabulary. Naturally, we then need to seek a compromise on the size of the models we use, as we cannot:

1. Use too large models during the research stage of the project, where we aim to process large corpora, evaluate the performance of the algorithms on them and make conclusions about the data. If we do aim to speed up this process, it is very likely that we would benefit from parallel computing — but processing large sizes of text in parallel has a non-negligible likelihood of running out of allocated memory even on some HPC clusters.

2. Force users to run too large models on their personal computers, as this would be a very poor user experience. We do not plan to hardcode any models (large or small) in the application, however, the provided guides will reference certain smaller-scale pretrained LLMs. Naturally, we would provide a way for more experienced and more capable organizations or individuals to run larger models with minimal effort.

4.2 Technology Choices

4.2.1 Python

We will be using Python version 3.10.X as shipped by the Anaconda software package. We are aware that Python 3.11 brings non-negligible optimizations and faster execution speed for some Python scripts, however, in light of the fact that the Anaconda distribution is still shipping Python 3.10.X, we will be using that version for the time being. We will be using the Anaconda distribution as it is a very popular and mature distribution of Python, which is also very easy to install and use. It also comes with a large number of pre-installed packages, which will be very useful for the current developer experience.

4.2.2 PyTorch

PyTorch "is a machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, originally developed by Meta AI and now part of the Linux Foundation umbrella. It is free and open-source software released under the modified BSD licence", as described by Wikipedia contributors (2023a).

Any models used in the application will be implemented in PyTorch, as it is a very popular framework for deep learning and natural language processing. It is also a very flexible framework, which allows for easy implementation of new models and algorithms. Furthermore, it is a very popular framework, which means that there is a large community of developers and researchers who have already implemented many of the algorithms we plan to use. This means that we can easily reuse their code and adapt it to our needs.

Comparison with TensorFlow

4.2.3 NLP Frameworks

NLTK

NLTK is a key Python library for natural language processing, primarily built for education purposes and managed as an open-source software, built to be relatively modular and lightweight. Commonly used by researchers and students for understanding and implementing algorithms for NLP tasks, it is a relatively popular and mature framework with a healthy extension ecosystem, where contributors are able to write their own modules and share them with the community.

NLTK

SpaCy

SpaCy is an open-source Python library for advanced natural language processing, designed to be easily used in production environments and implementing pipelines for enhanced NLP tasks. Whereas NLTK is primarily used for research and education, SpaCy is commonly being applied in industry environments.

Comparison between NLTK and SpaCy

Framework	Peak Memory	Increment
SpaCy	5089.13 MiB	4465.29 MiB
Mad Hatter	434.81 MiB	48.75 MiB

Table 4.1: <caption>

4.3 Code Style

4.3.1 PEP8

We will be using PEP8[1] (van Rossum et al., 2001) as our code style guide. PEP8 is a style guide for Python code, which is maintained by the Python Software Foundation. It is a very popular, and comprehensive style guide, widely used by many Python developers and organizations. It covers a wide range of topics, including naming conventions, indentation, line length, whitespace,

[1] `https://peps.python.org/pep-0008`

comments, “docstrings” (short for documentation strings, or, more specifically, comments that explain the way a given procedure or a class works, inside the code itself), and so on.

4.3.2 Docstrings

4.3.3 Linting

4.3.4 Testing

4.3.5 Code Review

4.3.6 Version Control

The outlined project

Git and GitHub

4.4 Documentation

4.4.1 Documentation Framework

We use Sphinx in this household.

4.4.2 Sphinx

4.4.3 Hosting

Chapter 5

Implementation

Chapter 6

Evaluation

In the context of application, we opt to implement several experiments that give a better understanding of the potential applications of MAD HATTER.

Component	Description
CPU	2.3 GHz Intel Core i9-9800H
RAM	16 GB DDR4 2400 MHz
GPU	Intel UHD Graphics 630 / Radeon Pro 560X
OS	macOS 13.1

Table 6.1: Specifications of the computer used for the experiments.

Wherever not explicitly mentioned, assume the specifications listed in Table 6.1.

6.1 Experimental Design

In this section, we describe the experiments we conducted to evaluate the performance of MAD HATTER. We start by describing the datasets we used for the experiments. Then, we describe the experiments we conducted and the metrics we used to evaluate the performance of MAD HATTER. Finally, we describe the baselines we used for comparison.

6.1.1 Datasets

Table 6.2 describes the utilized datasets along with their specific application in the experiment. Further descriptions of the datasets can be found at section 3.1.

Experiment	Dataset(s)
Document Class Identification	1. Project Gutenberg (PG) 2. EU DGT-Acquis & Europarl Corpus [NLTK] (Legal) 3. r/WRITINGPROMPTS (WP)
Authorship Identification	Project Gutenberg (PG) Up to 30 works from the 1000 most prolific authors
Machine-Generated Text Detection	1. WebText (representing real text) 2. Generated texts from GPT-2 XL-1542M

Table 6.2: Listing with the datasets used for the experiments.

[MK: note sometimes that gpt-2 texts are generated from the given training data]

6.2 Experiments

We implement three different experiments as a way of evaluating the performance of the application. The first experiment is a document class identification experiment, where we evaluate the performance of MAD HATTER in identifying the class of a document, thus demonstrating that the features we implement are well-defined and enable differentiating between different types of writing. We then move on to evaluating how well the algorithm can differentiate between different writing styles, a task also known as authorship identification. Finally, we follow the logical progression of authorship identification to address a topic that has been gaining traction in recent years, that of machine-generated text detection. This may have further applications in the future, as the field of natural language generation has been steadily growing in the past few years, with the advent of LLM such as GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020).

6.2.1 Document Class Identification

In this experiment, we evaluate the performance of MAD HATTER in identifying the class of a document. The datasets, described in Table 6.2, form the basis of the classes we designate, those being: (conventional) fictional literature (Project Gutenberg / PG), legal texts from the EU DGT-Acquis as well as the Europarliament Corpus distributed with NLTK (Legal / LG), and short-form stories from the subforum WRITINGPROMPTS of the social media platform REDDIT(WP).

Setup

Initially, all distinct texts are split into chunks of 100,000 characters (with the trailing chunk on its own). This is done primarily to maximize the potential data points of the dataset, but also to speed up the processing of the algorithm for large texts (for example, the texts in PG dataset are usually long-form full books which have upwards of 600,000 characters, assuming a ratio of 100,000 characters per 60-70 pages of text in traditional font and size). Normally, this may carry a potential for overfitting, as the chunks may not be representative of the whole dataset. However, as the texts are 1) very distinct from each other, and 2) have been shown to not split to more than 6-7 chunks, this is not a concern. The datasets are run through a simple pipeline that generates the features described in Section 3.3. For more flexibility in combining and comparing the datasets for classification, each dataset is separately run through the pipeline. After the features are extracted, each dataset is assigned its respective category. The datasets are then combined and shuffled.

The combined dataset is split into a training, validation, and test set, with a ratio of 80:10:10. The training set is used to train a logistic regression with L2 penalty, which is then used to predict the class of the documents in the test set. As an intermediary step, we run a grid search with the training dataset and the validation dataset in order to find the best parameter for the inverse of regularization strength of the algorithm. The parameter is chosen from the set $\{\frac{1}{64}, \frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8, 16, 32, 64\}$. The parameter with the highest accuracy on the validation set is chosen for the final model. The accuracy of the model is then evaluated on the test set.

Results

Table 6.3 shows the performance results for the document classification experiment. The results show that MAD HATTER is able to identify the class of a document with a very high accuracy. This is not surprising, as the classes are very distinct from each other, yet it affirms that the implemented features capture well specific characteristics of the text. The results also show that the model is

Table 6.3: Performance results for Document Classification

Experiment	Document Classification
Size of Train Set	4686
Train Accuracy	99.827%
Validation Accuracy	99.808%
Test Accuracy	99.827%

not overfitting, as the accuracy on the test set is very similar to the accuracy on the training set.

It should be noted that, despite the size of the training set is relatively small as opposed to other experiments in the field of document classification, the accuracy achieved is remarkably high. This is due to the fact that the features used are very simple and straightforward, and thus do not require a large amount of data to be learned. Furthermore, the algorithm is a step-up in terms of speed from existing baselines such as SVMs and TF-IDF algorithms, which makes it more suitable for large datasets and big scale text analysis. Figure 6.1 shows the confusion matrix for the document classification experiment. As seen, the document is able to distinguish between the classes with an excellent accuracy, precision and recall.

Discussion

Via the algorithm, the classes have been shown to not only be evidently distinct on their own, but also in terms of the features used. The features used in the experiment are very simple and straightforward, and thus do not require a large amount of data to be learned. Potential applications for document classification may include categorizing documents in a large database or potential dataset. Categorization can possibly be applied for sentiment evaluation for product reviews, social media posts, and so on. We go on to explore other potential uses of the algorithm in the following experiments.

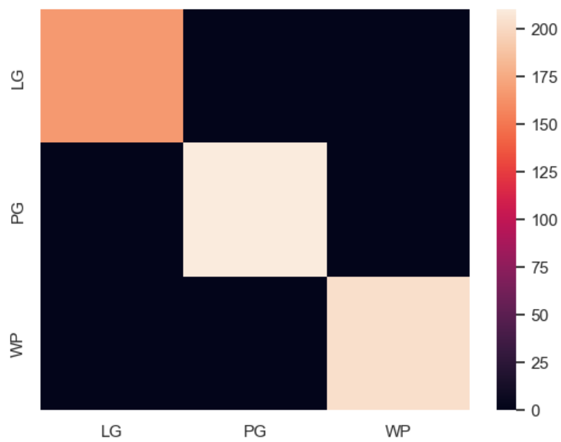


Figure 6.1: Confusion Matrix for Document Classification. The rows represent the true labels, while the columns represent the predicted labels.

6.2.2 Authorship Identification

After we have identified that the features are able to distinguish between different classes of documents, we now ask, “Can a machine distinguish between texts from the same class?”

This is a homogenous classification task, where the classes are very similar to each other, and the specific task is to identify the author of a text, given a set of candidate authors. For example, given the text of "Alice in Wonderland", MAD HATTER's task is to identify that the author is Lewis Carroll. The task is a natural progression from the document classification task and a more fine-grained existing problem in the field of NLP.

Setup

We make use of the work done by Gerlach and Font-Clos (2018) to standardize the Project Gutenberg for data exploration and analysis. We filter out for the most prolific 1000 authors in the available non-copyright literature. Furthermore, we randomly sample a maximum of 30 works per author. This is done in order to avoid overfitting for the more prolific authors (number one has more than 300), and even then only the top 200 authors have more than 30. A single chunk of 100,000 characters is then taken from each text and added to a list for processing.

The pipeline is similar to the one listed for document classification. We process all samples and obtain the features described in Section 3.3. The dataset is then split into a training, validation, and test set, with a ratio of 80:10:10. The features are then standardized (subtracting the mean and dividing by the standard deviation for each column). The training set is used to train a logistic regression with L2 penalty, which is then used to predict the author of the documents in the test set. As an intermediary step, we run a grid search with the training dataset and the validation dataset in order to find the best parameter for the inverse of regularization strength of the algorithm. The parameter is chosen from the set $\{\frac{1}{64}, \frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8, 16, 32, 64\}$. The parameter with the highest accuracy on the validation set is chosen for the final model. The accuracy of the model is then evaluated on the test set.

Experiment	A. Id. ($n = 1000$)	A. Id. ($n = 50$)
Size of Data (train/val/test)	17306:962:961	1290:72:72
Accuracy (train/val/test)	26.83%/23.91%/20.19%	55.50%/51.39%/56.94%
Precision	0.229/0.159/0.132	0.528/0.453/0.408
Recall	0.249/0.19/0.164	0.539/0.414/0.529
F1-Score	0.216/0.158/0.134	0.518/0.414/0.431

Table 6.4: Performance results for Authorship Identification ($n = 1000$ and $n = 50$).

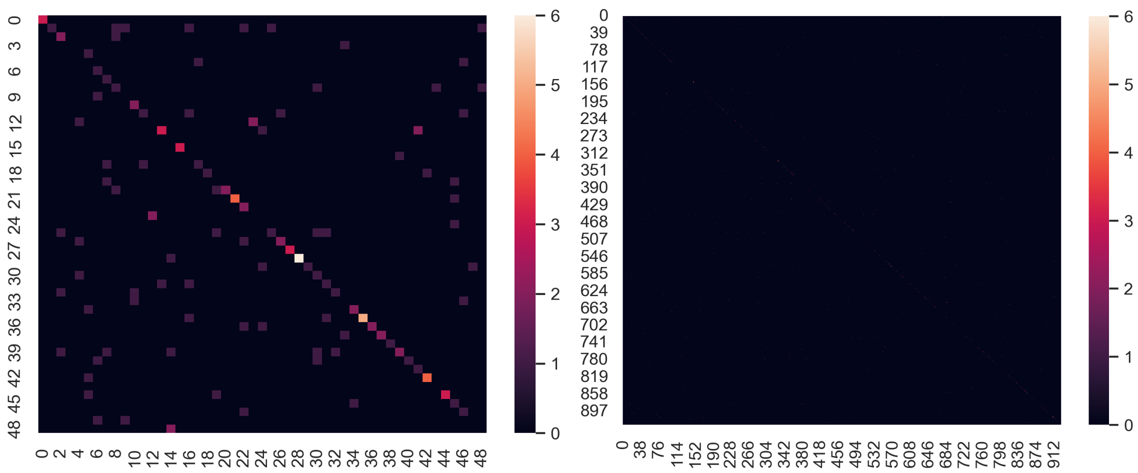
Results

Table 6.4 details the results of the case study for both the 50 authors case and the 1000 authors case. Accuracy, precision, recall and F1 scores are reported. The model is able to distinguish between authors with an accuracy of 55.50% for the 50 authors case, and 26.83% for the 1000 authors case. The results also show that the model is not overfitting, as the accuracy on the test set is similar to the accuracy on the training set. Although not clearly visible, Figure 6.2 shows the confusion matrix for the authorship identification experiment.

Unfortunately, the model struggles to achieve high accuracy for a huge number of authors, in our case the total number being a thousand. However, if we consider a baseline of a simple coin flipping, that is, a model that will randomly assign a label from the available labels (authors) to each work with a probability of $\frac{1}{1000}$ (for the bigger classification case), we can see that the model does perform relatively well in distinguishing some features that are similar between authors. The confusion matrix potentially indicates similar features between authors, a characteristic that may be a topic for further research. Note the highest number of correct predictions (6 and 6 out of a maximum of 30) respectively in the middle of the matrix in 6.2a and towards the tail end of the diagonal of 6.2b (around (855,855)). The other values are relatively low, with the highest number of incorrect predictions being 3 and 3, respectively.

The results are, in fact, in line with the results of Qian et al. (2019), who report an accuracy of 69.1% on the Reuters 50_50 (C50) dataset and 89.2 % on the Gutenberg dataset (for a maximum of 50 authors and 45000 paragraphs of text from their works). The authors used sentence- and article-level GRUs and an article-level LSTM neural network to achieve these results. The authors also provided a baseline accuracy of 12.24% via Gradient Boosting Classifier with 3 features, those being average word length, average sentence length, and Hapax Legomenon ratio (fraction of unique words), 2 of which we used. Given this baseline and the trivial computational complexity of our experiments, we have a reason to believe that we surpass the baseline and the researched features do enable stronger distinction between authors in this classification task.

Potential areas for improvement include the use of more sophisticated features, such as the ones described in **??**, as well as the use of more sophisticated models for multiclass classification, such as SVMs, neural networks, or Naive Bayes classifiers. We then move on to the next case study, which is the identification of machine-generated text.



(a) $n = 50$ **(b)** $n = 1000$

Figure 6.2: Confusion Matrices for Authorship Identification. The rows represent the true labels, while the columns represent the predicted labels.

6.2.3 Machine-Generated Text Identification

Having tested our hypothesis that the features are able to distinguish between different classes of documents in section 6.2.1, and then identified that the features are able to distinguish between different authors in section 6.2.2, we now move on to the ambitious goal of identifying machine-generated text. LLMs have seen explosive growth in the past few years, and generating human-like text, both grammatically and (somewhat) logically-sound, is now far from a distant dream. However, the ability to generate text indistinguishable from human-written text has raised concerns about the potential for misuse of such models. Particular issues may arise for example with academic grading, as AI writing tools become more and more prevalent. Fields other than academia may also be affected, such as journalism, where AI writing tools may be used to lazily generate non-proofread articles with the potential to spread misinformation. Of course, the potential for misuse is not limited to the above examples, and there are plenty of uses that malicious agents can come up with, either for personal gain or for the sake of spreading chaos.

Now then, we arrive back at the essence of the problem we are trying to solve: "What defines creativity? What defines human creativity?" The answer to this question is not simple, and it is not the goal of this thesis to answer it in full. However, we can attempt to answer a more specific question: "Can a machine distinguish between human-written text and machine-generated text?"

Setup

We make use of the WebText dataset (Radford et al., 2019), which is a large dataset of text scraped from the internet and used to train the influential GPT-2 (Generative Pretrained Transformer), a LLM developed by OpenAI. Texts generated by GPT-2 themselves serve as the basis for the "machine-generated" classification labels. Samples from the WebText dataset used to train GPT-2 form the basis for the "human-written" classification labels. 20,000 samples are randomly drawn from the set of machine-generated texts, and 20,000 — from the set of human texts.

The dataset is split into a training, validation, and test set, with a ratio of 80:10:10. The training set is used to train a logistic regression with L2 penalty, which is then used to predict the class of the documents in the test set. As an intermediary step, we run a grid search with the training dataset and the validation dataset in order to find the best parameter for the inverse of regularization strength of the algorithm. The parameter is chosen from the set $\{\frac{1}{64}, \frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8, 16, 32, 64\}$. The parameter with the highest accuracy on the validation set is chosen for the final model. The accuracy of the model is then evaluated on the test set.

Table 6.5: Performance results for MGT Detection

	Split		
	Train	Val	Test
Size of Data	32000	4000	4000
Accuracy	0.697	0.700	0.700
Precision	0.698	0.702	0.700
Recall	0.697	0.700	0.700
F1-Score	0.697	0.700	0.700

Table 6.6: Performance results for MGT Detection

			Split
	Train	Val	Test
Size of Data	32000	4000	4000
Accuracy	0.697	0.700	0.700
Precision	0.698	0.702	0.700
Recall	0.697	0.700	0.700
F1-Score	0.697	0.700	0.700

Table 6.7: Performance results for MGT Detection

		Size of Data	Accuracy	Precision	Recall	F1-Score
Split	Train	32000	0.697	0.698	0.697	0.697
	Val	4000	0.700	0.702	0.700	0.700
	Test	4000	0.700	0.700	0.700	0.700

Chapter 7

Discussion

7.1 Formal evaluation of the system

7.1.1 Accomplished goals

7.1.2 Limitations of the system

Bibliography

BNC Consortium (2007). British national corpus, XML edition. Literary and Linguistic Data Service.

Brants, T. (2000). Tnt: A statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, ANLC '00, page 224–231, USA. Association for Computational Linguistics.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. arXiv:2005.14165 [cs].

Brysbaert, M., Warriner, A. B., and Kuperman, V. (2014). Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46:904–911.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., and Zhang, Y. (2023). Sparks of artificial general intelligence: Early experiments with GPT-4.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. (2021). Evaluating large language models trained on code.

de Groot, A. M. (1989). Representational aspects of word imageability and word frequency as assessed through word association. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15(5):824.

Derczynski, L., Bontcheva, K., and Roberts, I. (2016). Broad Twitter corpus: A diverse named entity recognition resource. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1169–1179, Osaka, Japan. The COLING 2016 Organizing Committee.

Fan, A., Lewis, M., and Dauphin, Y. (2018). Hierarchical Neural Story Generation. arXiv:1805.04833 [cs].

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.

Franceschelli, G. and Musolesi, M. (2022). DeepCreativity: Measuring Creativity with Deep Learning Techniques. arXiv:2201.06118 [cs].

Francis, W. N. and Kucera, H. (1979). Brown corpus manual. *Letters to the Editor*, 5(2):7.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. (2021). The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027.

Gerlach, M. and Font-Clos, F. (2018). A standardized Project Gutenberg corpus for statistical analysis of natural language and quantitative linguistics. *CoRR*, abs/1812.08092.

Goldberg, Y., Adler, M., and Elhadad, M. (2008). Em can find pretty good hmm pos-taggers (when given a good start). In *Proceedings of ACL-08: HLT*, pages 746–754.

Hill, F., Reichart, R., and Korhonen, A. (2015). SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Leech, G., Rayson, P., and Wilson, A. (2014). Word frequencies in written and spoken English.

Manning, C. D. (2011). Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing: 12th International Conference, CICLing 2011, Tokyo, Japan, February 20-26, 2011. Proceedings, Part I 12*, pages 171–189. Springer.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs].

Mirowski, P., Mathewson, K. W., Pittman, J., and Evans, R. (2022). Co-Writing Screenplays and Theatre Scripts with Language Models: An Evaluation by Industry Professionals. arXiv:2209.14958 [cs].

Nigam, K., Lafferty, J., and McCallum, A. (1999). Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, volume 1, pages 61–67. Stockholom, Sweden.

Osgood, C. E., Suci, G. J., and Tannenbaum, P. H. (1957). *The measurement of meaning*. University of Illinois press.

Oxford English Dictionary Editorial (2023). Updates to the OED, March 2023. Blog.

Pierrehumbert, J. B. (2012). Burstiness of Verbs and Derived Nouns. In Santos, D., Lindén, K., and Ng'ang'a, W., editors, *Shall We Play the Festschrift Game? Essays on the Occasion of Lauri Carlson's 60th Birthday*, pages 99–115. Springer Berlin Heidelberg, Berlin, Heidelberg.

Qian, C., He, T., and Zhang, R. (2019). Deep Learning based Authorship Identification.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Schütze, H. and Singer, Y. (1994). Part-of-speech tagging using a variable memory markov model. In *32nd Annual Meeting of the Association for Computational Linguistics*, pages 181–187.

Thede, S. M. and Harper, M. (1999). A second-order hidden markov model for part-of-speech tagging. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 175–182.

Torralba, A. and Efros, A. A. (2011). Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528, Colorado Springs, CO, USA. IEEE.

van Rossum, G., Warsaw, B., and Coghlan, N. (2001). Style guide for Python code. PEP 8, Python Software Foundation.

Wikipedia contributors (2023a). Pytorch — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=PyTorch&oldid=1146375871`. [Online; accessed 2-April-2023].

Wikipedia contributors (2023b). Wordnet — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=WordNet&oldid=1143619785`. [Online; accessed 14-March-2023].