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

Abstract

An expansion of the title and contraction of the thesis.

Acknowledgements

Much stuff borrowed from elsewhere.

Contents

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, 2018a), 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>]

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

2.1.1 Part of Speech Tagging

2.1.2 Word Sense Disambiguation

2.1.3 Sentiment Analysis

2.1.4 Phonetic Analysis

2.1.5 Natural Language Generation

- top KP sampling
- challenges

2.2 Creative Measures

- 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

3.1 Datasets

Some introductory text on the purpose and uses of the datasets...

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 (2018b). 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

[1] `https://www.gutenberg.org/`

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.]

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.

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

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

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

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

Chapter 7

Discussion

Bibliography

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.

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.

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. (2018a). A standardized Project Gutenberg corpus for statistical analysis of natural language and quantitative linguistics. *CoRR*, abs/1812.08092.

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

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

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].

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

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.

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].