A systematic mapping review of algorithms for the detection of rhymes, from early digital humanities projects to the rise of large language models

Daniel G. Brown[1*], Rebecca Hutchinson[2], Carolyn E. Lamb[3]

[1] David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada

[2] Library, University of Waterloo, Waterloo, Ontario, Canada

[3] School of Computing, Queen's University, Kingston, Ontario, Canada

* Corresponding author

E-mail: dan.brown@uwaterloo.ca

Abstract

We survey fifty years of algorithms to discover rhymes in natural language text, focusing largely on rhymes in English, but also in Italic and other Germanic languages. Using a systematic mapping review, we filtered from 4704 initially reviewed studies down to 89 that were relevant to our research questions and satisfied our inclusion criteria. Older papers document the history of simple computer algorithms being used to analyze poetry, but these also include some that create text with rhyming patterns. Papers from 2006 to 2016 often include complex algorithms for teasing out complex rhyme definitions, particularly in the domain of rap music. More recent papers have moved to studying the use of large language models (LLMs) and either adapting their mathematical properties, or simply training them on a collection of rhyming text. We explore how grey literature (blogs, open-source programming projects and more) relates to the academic literature in rhyme detection, and we describe the complexity of engaging in systematic reviews of this sort in areas that span many disciplines.

Introduction

Algorithms for rhyme detection span the past half century, with researchers building computational systems for text analysis or generation of increasing sophistication during that time. In this study, we perform a systematic mapping review of research in this area, focusing on algorithms for Germanic (English, German, Norwegian) and Italic (Spanish, Portuguese, French, Italian, Latin, Romanian) languages. We categorize the time of creation, use case, data type, and variety of algorithms used in these studies, and map changes over time in the methods and uses of these algorithms. We investigate academic and grey-literature sources, and editorialize about our delight in the complexity of some

method, and the simplicity of others. We find everything from an early study by a Spanish literature PhD student studying a specific poet's rhyme patterns [1], in the 1980s, to co-creative, interactive poetry generation systems from the past year [2]. Rhyme detection is a window into both natural-language analysis and the use of computers in the digital humanities.

Why study rhyme algorithms?

We study the history of rhyme detection algorithms for a number of important reasons. First, they illustrate the history of change in computational analysis of text. Early authors mostly focused on simple analyses, though humanists in the field still tended to collaborate with computer scientists or used software developers to implement their ideas, as in Drake [1]. These methods are still used, but some of the simplest ones now are proposed as pedagogical examples (see, for example, Gries [3]). Later there was a period of complicated computer science algorithm development, where the barrier to entry from people outside computer science was high; other developers have needed to alter and update machine learning systems to search for the patterns they wanted to study. Now, in an era of ubiquitous machine learning algorithms, we see a collection of authors using state-of-the-art machine learning models without being forced to actually program them: they can simply train the models on a curated collection of poems or song lyrics, then use the ML models to classify and identify the properties they are looking for.

Similarly, these algorithms show off the changing methods for generation of new rhyming text. When the first author of this paper gave academic or public talks about master's student Hussein Hirjee's work [4,5] on detecting complex rhyme patterns in rap lyrics, members of the audience always asked, "what about generating new rap lyrics?" Later, Dekai Wu's research group in Hong Kong developed new algorithms to generate such lyrics [6,7] and Eric Malmi and his colleagues in Helsinki used ML methods

to predict which lines would best finish off a rap verse [8]. This work follows decades of work on generation of poetry, such as limerick algorithms [9].

Specific research questions

As we were not building a review toward the answer of a scientific question (as, say, in a medical study), our project's aims evolved as we examined the sources we found. These are some of our main research questions:

- What sorts of algorithms are used in these methods? Is it neural-network pattern finding, phonemic alignment, or more? To answer this question, we categorized each paper by the type of algorithm used.
- Does rhyme detection involve different approaches in different languages? How much work is done outside English? To answer these questions, we categorized papers by the language used.
- Does rhyme detection done by hobbyists look different from rhyme detection done by academic researchers? We analyzed academic and preprint literature in one group and other grey literature in a different group, and we present our findings below. We were not able to identify published papers by hobbyists, and many grey literature sources were from academics or students.
- Have the algorithms changed much over the past 10 years? Has there been a transition to machine learning approaches, or is there still a broad variety of methods?

Form of the review

We reviewed Grant and Booth's topology of reviews [10] and found that the goal of exploring the state of algorithms relating to rhyme would be best achieved by a systematic map of the literature. This choice is also supported by Kitchenham and Brereton [11], who compare and contrast systematic mapping reviews (SMR) with systematic literature reviews (SLR). SMRs focus on a topic area rather than try to answer a specific question; they provide an analysis of general trends in the research area. As this study is interested in the nature of the literature on algorithms relating to rhyme, rather than comparing the empirical results of studies, it falls firmly in the SMR category.

Methods

This study was guided by two frameworks. Petersen [12] gives recommendations for conducting SMRs. The PRISMA Extension for Scoping Reviews (PRISMA-ScR) Checklist and Explanation (2020) describes basic goals of scoping and mapping reviews [13]. Petersen's review identifies nine well used SMR guidelines which include Kitchenham [14], Arksey and O'Malley [15], Kitchenham and Charters [16], and Kitchenham and Brereton [11]. The Petersen review then consolidates processes and strategies across guidelines and provides updated best practice recommendations. We followed these updated guidelines along with PRISMA reporting guidelines. While scoping reviews and SMRs vary slightly, they both focus on a topic area rather than a answering a specific question [10]. This provides enough similarity to make the PRISMA ScR's reporting structure suitable for this study.

Eligibility

We wanted to consider a large collection of thematically-related papers, so we developed eligibility criteria with breadth in mind. However, we wanted to focus on papers that either fully described their rhyme detection algorithms, or focused on novel uses of those algorithms, rather than every paper that

uses rhyme detection methods. In particular, given that we did our literature search on November 21, 2022, there are probably dozens of papers that have come in the six months since then that use Chat-GPT (released on November 30, 2022) to build poetry, whose description would not have furthered our research outcomes. We discuss below how we filtered our data set to focus on primary sources of rhyme detection algorithms.

We wanted to focus on text analysis, so we excluded papers that used audio inputs only (such as recordings of songs or speeches). And, because we wanted to focus on definitions of "rhyme" familiar to the authors, we excluded papers focusing on Chinese or Arabic poetry or Hindi verse. To be consistent, we chose two language families, and allowed only papers with Germanic (English, Dutch, German, Norwegian) or Italic (Spanish, French, Portuguese, Romanian, Italian) language sources.

As such, we used the following criteria to identify studies relevant to the research questions:

- Inclusion
    - The research paper describes an algorithm for rhyme detection
    - The research paper gives proper reference to the algorithm that it uses in the poetry generation or analysis process
    - The algorithm is designed for use with natural language
    - The language for the writing is Germanic or Italic in origin
    - The algorithm is used with textual inputs/outputs
- Exclusion
    - The research paper is written in a language other than English
    - The algorithm is not designed for rhyme detection, or is insufficiently detailed, or the rhymes were not identified by an algorithm at all
    - Review papers

- The language for the writing is not Germanic or Italic in origin
- The algorithm is used for audio inputs/outputs only

Information sources

We searched eight literature databases on November 21, 2022 to find relevant literature for the research questions. The interdisciplinary nature of this topic necessitated the use of a variety of databases to find studies from computer science (ACM Digital Library), software engineering (IEEE Digital Library), computational linguistics (ACL Anthology), linguistics (Linguistics and Language Behavior Abstracts via ProQuest), music (Music Index via EBSCOhost), and literature (MLA International Bibliography and Communication & Mass Media Complete both via EBSCOhost). We also searched Scopus, as it is a large and multidisciplinary research database, and it has advanced searching capabilities which increase precision in systematic searching [17].

We considered, but decided against, three other databases. We did not include Google Scholar or SpringerLink, because an assessment of 28 databases for systematic searching assessed them as lacking the needed functionality for a primary review search [17]. We did not include ScienceDirect, because it, like Scopus, is an Elsevier product, and their overlap is significant.

We used citation searching, or snowballing, on the articles included for data extraction after the screening stage. Newer papers that cite included papers as well as the references of included papers were screened for inclusion. However, as we describe below, we strengthened the standards for inclusion during snowballing to avoid being overwhelmed with irrelevant studies.

In addition to the published literature, we conducted a grey literature search on January 30, 2023 to identify rhyming algorithms and approaches that may not have been written about in academic studies. Garousi, Felderer, and Mäntylä [18] and Zhang et al. [19] recommend a grey literature component when

the research questions rely on contextualizing the literature and/or if there is a large amount of practitioner created sources. This study's first author noted that some algorithms are developed outside of academia by hobbyists, so it was likely that there would be relevant information about these algorithms disseminated outside of the traditional research literature. These algorithms, and any documentation available on them, were considered for inclusion based on modified eligibility criteria. The algorithms must be used for rhyme detection, poetry generation, or poetry analysis; designed for natural language which is Germanic or Italic in origin; and used for textual inputs/outputs. We did not include items that merely cite a rhyme generator / detector. The grey literature source selection and search methods were modified from the those outlined by Godin et al. [20] and Garousi, Felderer, and Mäntylä [18]. Preliminary searching indicated that a Google search, preprint servers, and code repositories would be the best options for locating relevant items. arXiv https://arxiv.org/search/advanced, the premier preprint repository in computer science, and Humanities Core https://hcommons.org/deposits/ were searched for preprints and GitHub https://github.com/, Kaggle https://www.kaggle.com/, Pypi https://pypi.org/, and SourceForge https://sourceforge.net/ were searched for code.

Search

The database searches were developed in Scopus by the second author, who is a librarian with knowledge of systematic search techniques and the computer science literature. Librarians for music, English, and the digital humanities were consulted for database and search term suggestions. The Scopus search (displayed in Table 1) was reviewed by a health librarian with extensive expertise in systematic and scoping reviews and was used to develop the searches for the other databases (found in S1 Appendix). The final search results were exported from each database into Covidence https://www.covidence.org/ and any duplicates were removed.

Table 1. Scopus search, performed November 21, 2022.

Search	Query	Field	Results
1	music* OR song* OR poem* OR poet* OR lyric* OR verse*	Article title, Abstract, Keywords	371,754
2	troch* OR limerick* OR consonan* OR assonan* OR scansion* OR caden* OR iamb* OR prosod* OR sonnet* OR couplet* OR freestyl*	Article title, Abstract, Keywords	100,885
3	1 AND 2		5,344
4	sonnet* OR couplet* OR limerick* OR lyric*	Article title	5,722
5	{rhyme} OR {rhyming} OR {rhymes} OR stanza* OR quatrain* OR ballad*	Article title, Abstract, Keywords	9,696
6	3 OR 4 OR 5		19,498
7	automat* OR algorithm* OR comput* OR unsupervised OR un-supervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	Article title, Abstract, Keywords	12,647,518
8	6 AND 7		1,767
9	Limit 8 to English		1,677

The search development began by identifying the most important concepts within the research questions: algorithms and rhyme. However, we noticed through preliminary searching that not all papers fully describe their algorithm's function in the abstract, so the concept of rhyme was broken down further into two concepts: types of written items that often have rhyming and elements relating to rhyming structure. These concepts were nested within rhyme to create the search structure: algorithm AND (rhyme OR (types of written items that often have rhyming) AND (elements relating to rhyming structure)).

The subject matter expert (first author) provided relevant articles and the librarian (second author) selected appropriate keywords to add to the search. To improve recall, this search was implemented in

Scopus and results were sorted by relevance. More keywords were identified for inclusion in the search from the abstracts and author keywords of articles that looked highly relevant. This process was repeated in all the other databases except for the ACL Anthology, due to its lack of advanced search functionality. Ideas for keywords were also found in the rhyme entry in the Encyclopedia of Language & Linguistics [21] as well as from the controlled vocabulary in the Linguistics and Language Behavior Abstracts ([22] and Communication & Mass Media Complete ([23] databases.

The resulting search provided an unmanageable number of papers for review. To improve precision, keywords were tested in Scopus to determine how many relevant articles would be missed by removing the term. Sorting by relevance, the first 20 results were examined. If none of the results looked obviously relevant that term was removed. If one or more looked relevant, then we tried to retain the term. Some terms returned hundreds to thousands of irrelevant results, so the term had to be removed to ensure screening remained manageable. In those cases, a more specific keyword or key phrase was identified to find the relevant papers without bringing back so many of those irrelevant ones. Due to the lack of descriptive terms in the abstracts for this topic there were a few times this was not successful.

In some cases, the search structure was modified to improve precision. For example, in addition to "limerick" and "sonnet" finding articles about these poetic forms, they also found many articles about research at the University of Limerick and research using a variety of software or systems called Sonnet (either from a company of that name, or as the name of a software system [24]). Therefore, in some databases, these terms had to be searched in the title only or combined with other terms relating to rhyming structure. Another example is the homophone words "rhyme" and "rime." "Rime" is an alternate spelling of rhyme, so if not restricted by quotation marks, Scopus automatically searches for "rime" as well. Unfortunately for the search, "rime" also refers to the buildup of ice [25]. Searches that did not remove this spelling brought in an overwhelming number of irrelevant engineering articles, so

206 “rhyme” had to be searched in braces to prevent Scopus from expanding the search to include its
207 homophone.

208 The search structure was also adapted as needed to account for the difference in database content.
209 Since the ACM, IEEE, and ACL databases are already focused on computer science publications, it was
210 unnecessary to search them for the algorithms concept; similarly, the content of the Music Index made it
211 unnecessary to search for the types of written items with the rhyming concept. Leaving out these
212 concepts allowed the search to retrieve items that may have been missed if those concepts lacked our
213 chosen keywords.

214 We needed a workaround for the ACL Anthology because its website search uses Google search, and as
215 such was unable to use the keywords and Boolean structure of our queries as we intended. While ACL
216 Anthology does not have a method for batch exporting search results, it does allow one to batch
217 download the full anthology as BibTeX with abstracts; the first author wrote a custom Python script to
218 filter these records for those that matched our complex search criteria.

219 Due to the lack of descriptive terms in some abstracts and the number of noise-generating terms that
220 had to be searched in a more limited manner, this search may not have found all relevant literature. We
221 augmented our search results by adding the references in the bibliographies (taken from Scopus) of
222 included papers, and by using Google Scholar to identify papers citing the ones we had included.
223 However, in this snowballing phase, we did not include papers that merely use existing rhyme algorithms
224 as subroutines in the rhyme generation process, as we would have then wound up with all papers that
225 automatically generate rhyming poetry. Having a few such papers from the first phase of the process
226 was fine, as it gave us a sense for what the use of rhyme detection algorithms is, but we would otherwise
227 have been swamped with papers that do not answer our research questions.

Simple searches were necessary for the preprint servers, code repositories, and Google because their search interfaces generally lack advanced features. Because of this limitation, a small subset of the database keywords was used for each search, focusing on the ones which found only the most relevant results. This subset was derived from multiple preliminary searches of the grey literature sources as suggested by Garousi, Felderer, and Mäntylä [18] and Zhang et al. [19]. This worked well for arXiv, a STEM-focused repository: the rhyming concept was searched in the title or abstract fields, resulting in a reasonable number of results. Humanities Core required a search including both algorithm and rhyming concepts, but it had difficulty with the Boolean structure which meant a variety of phrases had to be searched separately. This was also true for GitHub. Originally, we thought that the code repository searches would only require the rhyming concept, which worked in Kaggle, Pypi, and SourceForge, but this led to an unmanageable number of results in GitHub. GitHub also had difficulty with the Boolean structure and so a variety of phrases had to be searched separately.

No custom Google search engines or advanced search options were identified that would benefit this search so multiple simply structured searches containing both concepts were conducted to find relevant results. We saw no evidence with our Google searches that using incognito search improved search outcomes, so we did not use this mode.

Selection of sources of evidence

Database results were imported to Covidence, which identified and removed duplicates. The first author, an expert in algorithmic rhyme detection, analysis and generation (see, for example Lamb et al. [26] and Sawicki *et al*. [27–29]), screened the titles and abstracts for relevancy based on the eligibility criteria. The first and third authors, both experts on computer-generated poetry, then screened each full text for eligibility and any disagreements were resolved through discussion between the two reviewers. The same methods were used to identify relevant articles from the citation searching, except that we did

not include articles that only used rhyme detection algorithms from other works as subroutines in poetry generation.

It was not possible to batch export from the grey literature sources so items were screened on the results pages and relevant ones were exported to Zotero. Any preprints or research papers found through the grey literature searches were moved to Covidence for removal of duplicates and screened with the results from the citation searching.

In addition to the inclusion and exclusion criteria, the GitHub repository results were only considered if they are public and have at least one positive rating. The Google results were considered until the reviewer reached an entire page (10 or more results) that did not look relevant. Eligible items were added to a Zotero folder and duplicates removed.

Data charting and synthesis process

We assessed each included study for a number of specific variables, in part derived from our study goals.

We characterized each paper by its period of creation: “Early”, meaning before 2006; “mid”, meaning between 2007 and 2016; and “late”, or after 2016.

We determined whether the goal of a study was to analyze rhyming text, to generate new rhymes, to co-create rhymes with humans, or something else.

We divided the data sets used in these papers into music lyrics, poetry, both, or something else; for the lyrics, we also identified whether they were rap lyrics, since as we note below, this genre has been particularly productive for development of rhyme detection methods.

We described whether the paper’s rhyme detection algorithm was fully specified, and if so, which of the following describes it: a simple algorithm based on matching vowel phonemes; a complex algorithm using sequences of phonemes; a tweak to modern machine learning algorithms; a reliance on a language

model; or something else. If an algorithm was not fully described, but was included by citation, we identified which paper was the algorithm's source.

Finally, for each paper, we identified the natural language or languages from which the data or generation comes. This category we did not fill during the extraction phase, but instead the first author extracted it during the paper-drafting stage.

The data was charted by the first author for all studies. Studies were included without critical appraisal as the study's goal was to characterize the literature rather than evaluate it.

Several manuscripts fit multiple categories,. For example, a paper might include both simple phonemic tricks and ML tricks. If so, we included both tags; when we build the charts in later sections, we have simplified the outcomes. Similarly, a few papers were extremely vague about what their algorithm was, which made analysis difficult. We had considered identifying the role of creators of research papers (students or professors, for example), but most papers did not include this information, so we dropped it, except for the paper describing the Wall Street Journal package about the rhymes in *Hamilton* [30].

Results and Discussion

Selection of sources of evidence

All database search results totaling 3594 items were exported into Zotero and then into Covidence. Duplicates were removed in Covidence and 2970 unique publications were screened. 126 publications were identified for full text review and of those 68 were included for data extraction and analysis. One article, "Using Computers to Analyze Poems: A Phonological Metrification" [31] was excluded because the full text could not be found. 57 were excluded by the reviewers for not matching the inclusion criteria (see the PRISMA Diagram, Fig 1, for more detail.) Using the same methods, 1722 unique articles from citation searching were screened and 33 were identified for full text review. 17 were excluded,

again, for not matching the goals of the study. As noted above, we did not include papers found by snowball sampling that simply used a rhyme detection algorithm as a component of poetry generation. 12 unique items from the grey literature searches were preprints from arXiv or research papers found through Google. 10 unique items were added to full text review and 5 were excluded for not matching the study's goals. Lastly, 47 code repository results and non-research paper results from Google were included for review.

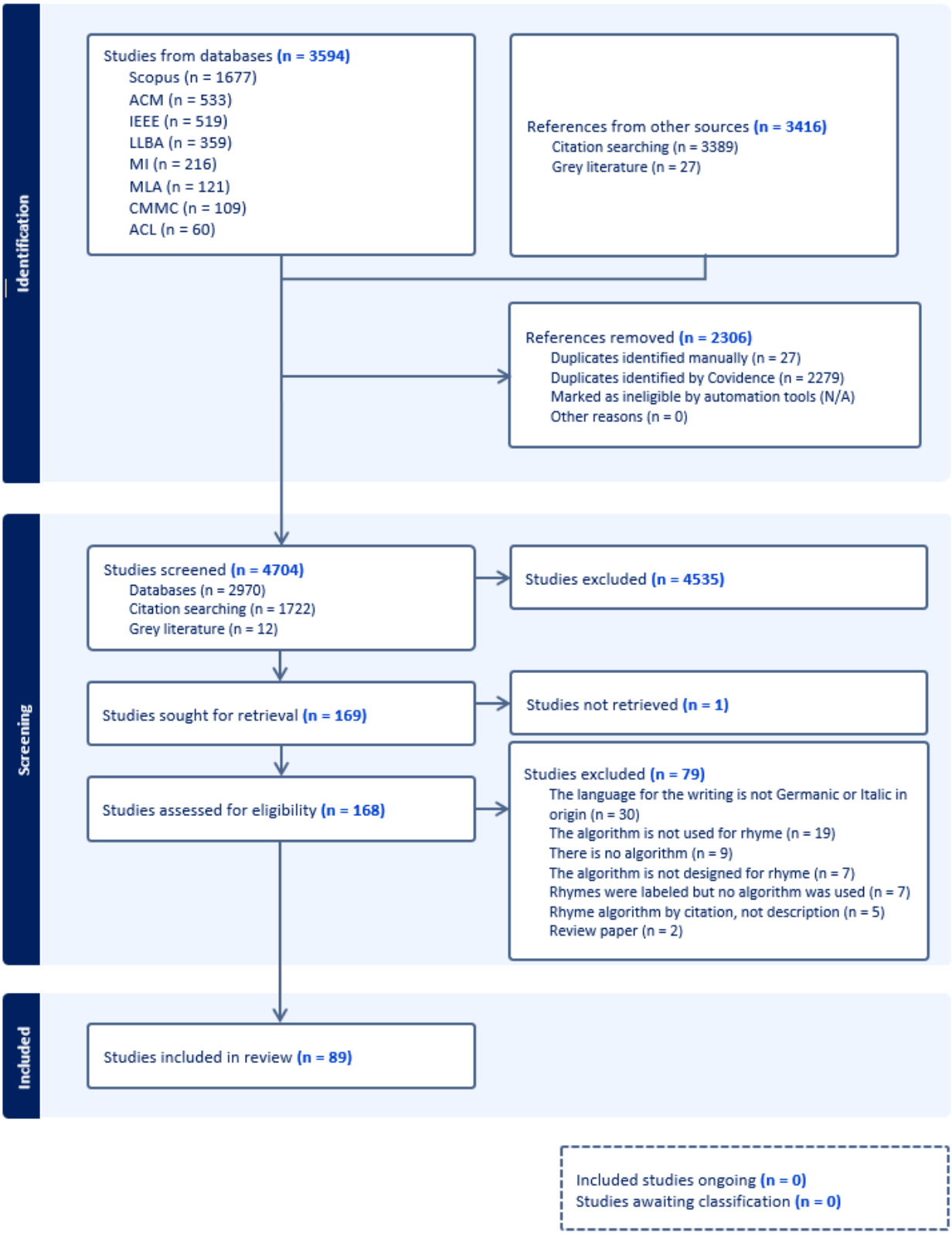

Fig 1. PRISMA Diagram

Findings

At the end of our filtration procedure, we had 89 manuscripts to extract, which can be seen in Table 2, in addition to 47 grey-literature objects (mostly open source software projects and popular press articles, but also blogs).

Table 2. Included articles' extraction table

Study	Era	Purpose	Data type	Techniques used	Language
[32]	Middle	Analysis	Music lyrics (rap)	ML tweaks	English
[33]	Middle	Generation	Music lyrics (rap)	From another paper	English
[34]	Late	Generation	Poetry	ML tweaks	English
[35]	Middle	Generation	Music lyrics (rap)	ML tweaks	English
[36]	Middle	Generation	Music lyrics (not rap)	Hard-coded phoneme/letter rules	English
[37]	Late	Generation	Poetry	Complex alg (alignment, scoring rules)	English
[38]	Late	Analysis	Poetry	Complex alg (alignment, scoring rules)	English
[39]	Late	Generation	Poetry	model-reliant :The NN just learns	English
[40]	Late	Generation	Poetry	From another paper	English
[41]	Late	Generation	Music lyrics (rap)	Hard-coded phoneme/letter rules; ML tweaks	Spanish
[42]	Late	Generation	Music lyrics (not rap)	Complex alg (alignment, scoring rules)	English
[43]	Late	Generation	Music lyrics (not rap)	Hard-coded phoneme/letter rules	English
[44]	Early	Other	Other	Hard-coded phoneme/letter rules	English
[45]	Late	Generation	Other	Hard-coded phoneme/letter rules	English
[46]	Late	Generation	Music lyrics (not rap)	Hard-coded phoneme/letter rules	English
[47]	Middle	Analysis	Poetry	Hard-coded phoneme/letter rules	Romanian
[48]	Middle	Generation	Poetry	Hard-coded phoneme/letter rules	English

[49]	Middle	Analysis	Poetry	Hard-coded phoneme/letter rules; Complex alg (alignment, scoring rules)	English, Italian
[1]	Early	Analysis	Poetry	Hard-coded phoneme/letter rules	Spanish
[50]	Middle	Generation	Poetry	Hard-coded phoneme/letter rules	English
[51]	Late	Generation	Poetry	Hard-coded phoneme/letter rules; Complex alg (alignment, scoring rules)	Norwegian
[52]	Middle	Generation	Poetry	Other	English
[53]	Late	Generation	Music lyrics (rap)	model-reliant :The NN just learns	English
[54]	Late	Analysis	Poetry	Complex alg (alignment, scoring rules)	English
[55]	Late	Generation	Poetry	Hard-coded phoneme/letter rules	Romanian
[56]	Late	Generation	Music lyrics (not rap)	Hard-coded phoneme/letter rules	English
[57]	Late	Generation	Music lyrics (not rap)	Hard-coded phoneme/letter rules	English
[58]	Middle	Generation	Poetry	Other	English, French
[59]	Late	Generation	Poetry	Other	Spanish
[60]	Middle	Generation	Poetry	Hard-coded phoneme/letter rules	English
[61]	Late	Generation	Poetry	From another paper	English
[62]	Middle	Analysis	Poetry	Hard-coded phoneme/letter rules	English, Italian
[3]	Late	Other	Poetry	Hard-coded phoneme/letter rules	English
[63]	Late	Analysis	Music lyrics (not rap); Poetry	model-reliant :The NN just learns	English, German, French
[64]	Late	Generation	Poetry	Other	English
[65]	Late	Analysis	Music lyrics (not rap)	From another paper	Spanish
[66]	Late	Generation	Music lyrics (rap)	Hard-coded phoneme/letter rules	English
[30]	Middle	Analysis	Music lyrics (rap)	Complex alg (alignment, scoring rules)	English

[67]	Middle	Analysis	Music lyrics (rap)	Complex alg (alignment, scoring rules)	English
[68]	Middle	Analysis	Music lyrics (rap)	Complex alg (alignment, scoring rules)	English
[4]	Middle	Analysis	Music lyrics (rap)	From another paper	English
[5]	Middle	Analysis	Music lyrics (rap)	Complex alg (alignment, scoring rules)	English
[69]	Late	Generation	Poetry	ML tweaks	English
[70]	Late	Generation	Music lyrics (rap)	Hard-coded phoneme/letter rules	English
[71]	Late	Generation	Poetry	ML tweaks	English
[72]	Late	Generation	Poetry	ML tweaks	English
[9]	Early	Generation	Poetry	Hard-coded phoneme/letter rules	English
[73]	Late	Generation	Music lyrics (not rap); Poetry	ML tweaks	English, Chinese
[74]	Late	Analysis	Music lyrics (rap)	ML tweaks	English
[75]	Late	Analysis	Music lyrics (rap)	Complex alg (alignment, scoring rules); model-reliant :The NN just learns	English
[76]	Late	Generation	Poetry	model-reliant :The NN just learns	English
[77]	Late	Analysis	Music lyrics (not rap)	Hard-coded phoneme/letter rules	English
[8]	Middle	Generation	Music lyrics (rap)	Hard-coded phoneme/letter rules	English
[78]	Late	Generation	Music lyrics (rap)	ML tweaks	English
[79]	Late	Analysis	Poetry	Hard-coded phoneme/letter rules	Spanish
[80]	Late	Analysis	Music lyrics (not rap)	From another paper	English
[81]	Late	Analysis	Poetry	Hard-coded phoneme/letter rules	Portuguese
[82]	Late	Generation	Music lyrics (rap)	ML tweaks	English
[83]	Late	Analysis	Poetry	Complex alg (alignment, scoring rules)	Latin
[84]	Late	Generation	Music lyrics (rap)	ML tweaks	English
[85]	Late	Generation	Poetry	From another paper	English, Portuguese
[86]	Late	Co-creativity	Poetry	From another paper	English, Portuguese

[87]	Late	Generation	Poetry	Hard-coded phoneme/letter rules	English, Portuguese
[88]	Late	Co-creativity	Poetry	From another paper	English, Portuguese
[89]	Late	Generation	Poetry	model-reliant :The NN just learns	English
[90]	Middle	Analysis	Poetry	Hard-coded phoneme/letter rules	English
[91]	Late	Analysis	Poetry	Complex alg (alignment, scoring rules)	English, Czech, French
[2]	Late	Co-creativity	Poetry	ML tweaks	French
[92]	Middle	Generation	Music lyrics (rap)	model-reliant :The NN just learns	English
[93]	Middle	Analysis	Music lyrics (rap)	From another paper	English
[94]	Late	Generation	Music lyrics (not rap)	From another paper	English
[95]	Late	Generation	Poetry	model-reliant :The NN just learns	English
[96]	Middle	Analysis	Poetry	Hard-coded phoneme/letter rules	English
[97]	Early	Analysis	Poetry	Hard-coded phoneme/letter rules	Italian
[98]	Late	Generation	Music lyrics (rap)	Hard-coded phoneme/letter rules	English
[99]	Middle	Analysis	Music lyrics (not rap); Poetry	From another paper	English
[100]	Middle	Analysis	Poetry	Complex alg (alignment, scoring rules)	English
[101]	Late	Generation	Poetry	ML tweaks	English
[102]	Late	Generation	Poetry	From another paper	English
[103]	Early	Generation	Poetry	Hard-coded phoneme/letter rules	English, Bulgarian
[104]	Late	Generation	Poetry	From another paper	English
[105]	Late	Generation	Poetry	ML tweaks	English
[106]	Early	Generation	Poetry	From another paper	unknown
[7]	Middle	Generation	Music lyrics (rap)	ML tweaks	English
[6]	Middle	Generation	Music lyrics (rap)	Complex alg (alignment, scoring rules)	English
[107]	Middle	Generation	Music lyrics (rap)	ML tweaks	English
[108]	Middle	Generation	Music lyrics (rap)	From another paper	English

[109]	Late	Generation	Poetry	ML tweaks	Italian
[110]	Late	Generation	Poetry	ML tweaks	English

Typology

Table 3 divides our 89 manuscripts into three eras of “early”, “middle” and “late” based on their publication date, and classifies whether the manuscripts are about generation of new rhymes or analysis of existing ones. In Table 4, we also show whether papers are about music or poetry or a different kind of data, and we highlight which papers are in particular about rap or hip hop lyrics.

Table 3. Primary purpose of algorithms for rhyme detection. Later papers largely concern themselves with generating new rhyming text.

Purpose	**early: 1960s to 2005**	**mid: 2005 to 2016**	**late: 2017 to present**	**Grand Total**
Analysis	2	14	12	28
Co-creativity			3	3
Generation	3	14	39	56
Other	1		1	2
Grand Total	**6**	**28**	**55**	**89**

Table 4. Data type used in rhyme detection. Early papers focused on poetry, while later papers pivoted to studying music lyrics as well. In the middle period, the complex rhyme patterns found in rap music were a particularly useful focus for researchers interested in applying computer science methods from other domains to the rhyme detection task.

Data type	**early: 1960s to 2005**	**mid: 2005 to 2016**	**late: 2017 to present**	**Grand Total**
Music lyrics: not rap		1	9	10
Music lyrics: rap		15	10	25
Music; Poetry		1	2	3
Poetry	5	11	33	49
Other	1		1	2
Grand Total	**6**	**28**	**55**	**89**

Finally, in Table 5, we describe the kinds of algorithms people have developed: for this, we only consider the papers that describe rhyme detection or generation algorithms, rather than using other sources and citing them. (Despite this restriction, we still may include the same algorithm multiple times: for example, the algorithm by Hirjee and Brown appears in increasingly full detail in an ISMIR conference paper [67], in an *Empirical Musicology Journal* paper [5], and in Hussein Hirjee's MMath thesis from the University of Waterloo [68]). We describe several types of rhyme detection algorithms. Some algorithms use simple phonemic rules (two words rhyme if their last vowels and any subsequent consonants are the same, for example). This category is particularly common in earlier periods, and works especially well for languages like Italian [97] or Spanish [1].

Table 5. Algorithm variety used in rhyme detection. Early papers use simple rules, which are widely present in later papers as well. Complex algorithms become common in the middle period, while late algortihms include more machine learning approaches. Papers that use more than one method are mapped to a single category: approaches that use complex algorithms and hard-coded phoneme rules are mapped to the complex category. Note that the 16 papers that used an external rhyme mapping algorithm are not described here, which is why the grand total is 73 papers.

Algorithm type	early: 1960s to 2005	mid: 2005 to 2016	late: 2017 to present	Grand Total
Complex		7	8	15
Hard-coded rules	5	9	14	28
ML tweaks		4	15	19
The model just learns		1	6	7
Other		2	2	4
Grand Total	**5**	**23**	**45**	**73**

Some authors, beginning with Hirjee and Brown [4,5,67], provide complex algorithms that highlight imperfect rhymes or rhymes that cross syllables. These algorithms tend to compute alignments of syllable phonemes, using either hand-created scoring schemes or ones that are learned automatically.

Another group of algorithms is based on changes to machine-learning algorithms for text generation: by altering the output probability distribution of an ML model, these systems can force rhyming constraints

onto the generated text, essentially conditioning the model to only output text that satisfies rhyme constraints. Typically, rhyme is defined in these cases are simple phonemic rules, but the process is challenging since the ML model normally is concerned with words, not syllables or phonemes.

Finally, a small number of papers, starting in 2016, retrain or fine-tune a language model on a data set of rhyming texts, and expect that this process will allow the language model to learn the rhyme constraints of a form of poetry from its own information. This post-algorithm-design phase may become much more common in the future.

Assigning manuscripts to these categories is not always simple, but we have done our best to use this simplified taxonomy to present our information. In the next section, we focus on the trends during the three eras and give examples. Then we discuss differences between generation and analysis, and we give a broader discussion of the types of algorithms under discussion. Finally, we note the significance of rap music in developing this field, and briefly discuss some sociopolitical aspects of this connection to a specific, somewhat stigmatized ,musical genre.

Era of algorithms

We divided our sources into three periods: “early” papers from before 2006, “middle” papers from 2006-2016, and “late” papers from 2017 to when we did our search.

Early papers are the six pioneering papers before 2006. Because of incomplete digital archives, we only have abstracts or very short manuscripts for many of these papers. All but one focus on poetry, with three on poetry generation and two on poetry analysis.

The earliest fully-described algorithm in our data set is in the PhD thesis of Dr. Mildred Murphy Drake [1], which presents her research as a graduate student in the department of Spanish and Portuguese at the University of Colorado. Dr. Drake’s thesis is about automatic detection of internal rhymes in the *poesía*

desnuda of Juan Ramón Jiménez, and it uses simple vowel rules to identify whether two words rhyme. In 1994, the same author published a 1-page review of a HyperCard-based multimedia library for teaching about Latin-American poetry [111]. Many parts of Dr. Drake's thesis, despite being from 1987, could have been written very recently. She is prescient in planning for a world in which AI methods are primarily predictive, not analytic: "The A.I. promoters will be depending on the more predictable qualities of human language in order to represent the achievements of cognitive science in producing computer-generated decisions and 'ideas'."

Other early papers largely use simple phonemic rules, or reference an existing rhyme dictionary: "X rhymes with Y if the dictionary says so." Simple phonemic rules work especially well for languages like Spanish, where spelling indicates stress and phonemic content much more easily than in English. In English, only a phonemic transcription allows us to know that, for example, "rough" does not rhyme with "through". Byrd and Chodorow's 1985 paper, "Using an on-line dictionary to find rhyming words and pronunciations for unknown words," talks about using a phonemic dictionary, and then describes some rules for identifying which phonemic patterns are closer rhymes and which are not [44].

Lessard and Levison presented a limerick generator in 2005 which uses hard-coded rhyme rules in English [9]. In many ways, their method resembles some later constraint-propagation approaches; they phonologically represent words, then pre-identify the closing phoneme patterns for the rhyming lines, along with the meter of the lines and templates that implement customary rules of how a limerick works ("There once was a [person] from [place]", etc.)

Middle papers are those that appeared between 2006 and 2016, and tended to focus on poetry or lyric analysis (though some include generation as well). Many of these papers use simple heuristics to identify whether two words or two lines rhyme; eight still use straightforward phonemic tricks.

One clever paper from 2011, by Reddy and Knight, uses a simple phonemic description for what rhymes with what (looking only at last words, focusing only on final stressed vowels and subsequent syllables), but focuses on identifying rhyme schemes, using machine learning methods like expectation-maximization to identify which of a collection of rhyme schemes is most likely to apply to a given poem [96]. Similarly, the heavily cited DopeLearning paper by Malmi *et al*. [8] selects next lines from a database of rap music lines, using rhyme information (derived from looking for sequences of matching vowels) and other text data to determine how best to continue a rap song.

The work of Hirjee and Brown (presented across three manuscripts: a conference paper [67], an archival journal paper [5], and Hirjee's master's thesis [68]) uses a more complex algorithm, which identifies imperfect and internal rhyme in rap lyrics; it uses a scoring algorithm and a bioinformatics-inspired phonemic sequence alignment system to identify words which are more likely to come from rhyming text than from a null model of random syllables. Their algorithm is used by a number of later authors in analysis and generation of text; both the work of Potash et al. [92,93] on language ghostwriting and the work of Singhi and Brown [99] on analysis of music lyrics use the Hirjee/Brown method.

The general idea of this work (allowing for multiple syllables and words to rhyme, incorporating not-quite-exact matches, and handling syllables with different numbers of phonemes in them) is also found in some other work from this period. Hinton and Eastwood [30] came up with a simpler scoring in analyzing the rhymes in the hit Broadway musical "Hamilton", as do Tanasescu, Paget and Inkpen [100] in their work analyzing meter and rhyme of poetry. The Hinton and Eastwood paper [30] describes an astonishingly effective data visualization package at the Wall Street Journal about Lin Manuel Miranda's amazing rhymes; it is also the only paper in our extracted data created primarily by journalists.

Other authors in this period came up with different machine-learning ideas that incorporate rhyme. Wu, Addanaki and Saers [6] built a Hidden Markov Model (HMM)-based system to identify rhymes in

phonemic transcripts, where their system identifies rhyme schemes, and allows for poly-syllabic rhymes cleverly. They use their own rhyme discovery approach in a sequence of papers on building a rap battler bot [6,7,107,108], and their focus on rap requires them to consider imperfect rhyme.

Middle papers are evenly split between fourteen papers with a focus on generating rhymes and fourteen with a focus on analysis. Papers about generation use complex rules for creating rap rhymes. For example, the group centred around Dekai Wu largely uses the Addanki and Wu [32] algorithm based on HMMs, where each state of the HMM corresponds to a rhyme scheme, and generation or analysis consists of either sampling from the HMM or finding a maximum-probability explanation of a sequence of words given a HMM structure. This method does not allow for internal rhymes.

Another outlier from this period is the pair of papers by Potash, Romanov and Rumshimsky [92,93], which simply train a long short-term memory (LSTM) network on a large corpus of data and attempts to have it learn to make rhyming rap verses. To the best of our knowledge, this is the first project of this sort; all other papers using this "just train it" approach come from the late period.

Late-period papers and the rise of machine learning. More than half of the studies we reviewed (55 of 89) came since 2017. 2017 is the year when Google first announced the transformer language model [112], which gave rise to the astonishing flowering of AI language models of the early 2020s. As such, we see a huge change in both the subject of papers that include rhyme detection (since more of them are now trying to constrain language models to create rhyming text) and also in the methods used by such models. We also start to see far more focus on languages outside English. (However, we note that our initial decision to restrict to Germanic and Italic languages may have excluded earlier studies of rhyme in languages like Chinese, Finnish, Basque, Russian, or Arabic.)

The majority (39 out of 55) of late-period papers cover generation, not analysis. For example, Chang et al. [46] uses rhyming as part of its overall task of composing singable lyrics, while the well-cited Deep-

speare paper [72] builds poems in sonnet form. Several papers from this period automatically produce rap lyrics. This set also includes the well-cited Hafez paper [61], which built upon earlier work by the authors [60] in building high-quality poems using recurrent neural networks.

In this later period, a small number of works do use pre-existing algorithms; for example, the Hirjee and Brown methods [4,5] are used by Uthus et al. [102] and Melvill-Smith et al. [80]. Some more mature systems for poetry generation like PoeTryMe have been published in multiple venues and continue to use their original rhyme algorithms [85–88].

Sixteen papers from the late period still use simple phonemic rules to identify when lines rhyme; some of these are for languages where rhyme is easier to describe than in English. A similar number use machine-learning tweaks to build rhymes by altering the distribution probabilities of the output to ensure that only words with matching vowel phonemes are generated, for example Hopkins [69].

Six papers from the late period (and only one other from the middle period, the Potash et al. papers cited above [92,93]) use a tuned natural language model as the sole source of rhyme information. That is, rather than giving a definition of rhyme or an algorithm for rhyme detection, these papers give training data that includes rhyme to a language model which implicitly learns about rhyme by parroting properties of the input distribution; the model can then create new rhymes. These methods work to differing degrees; for example, GPoeT-2 [76] generates limericks that are not especially coherent, but which do rhyme successfully. These methods will surely become more common in the near future, given that large language models released since we did our search are more successful than earlier ones: GPT-2, even when fine tuned, does not produce nearly as good poetry as does a fine-tuned GPT-3 model [29].

Eight papers from the late period describe complex algorithms, typically incorporating alignment of phonemes with nontrivial scoring rules. Bay, Bodily and Ventura [38] give scores based on phonemic properties for consonants and based on distance in a Euclidean embedding of the IPA English Vowel

Chart for vowels. Plecháč [91] builds a probability-scoring approach using co-frequency of paired phonemes in known rhymes, compared to the baseline frequency of those phonemes (which is similar to Hirjee and Brown [4,5]). Similarly, Nagy [83] scores syllable rhymes using a self-created vowel-scoring matrix, then identifies stanzas of text with large numbers of rhymes, assuming a null model of random lines of Latin text.

An extreme outlier is the Ferraz de Arruda et al. [54] paper from a physics journal, which uses digital signal processing techniques to identify patterns in phonemic sequences; it's not obvious how to map these techniques onto pre-existing algorithms, and the paper largely does not cite those previous works.

Another outlier from this time is the 2022 master's thesis of Tita Enstad, from the University of Oslo [51]. This author uses a phonemic alignment method, but she focuses on questions of dialect in identifying which Norwegian words rhyme; in particular, the goal of this project is to find rhymes in modern Norwegian as segregated from Danish. As such, Enstad spends a fair amount of time distinguishing Norwegian words derived from Danish, and she focuses on poetic rhymes that are from demonstrably Norwegian texts. In that sense, this author's work is simple phonemically, but complex politically.

Types of algorithms

Overall, we identified a wide variety of algorithms used to detect and generate rhymes. Early algorithms tended to use simple tricks based on phonemic dictionaries or on simple definitions of "syllable" and "rhyme", an approach still used by some in the present day. One example is a sample homework assignment for a first-year computer science course [3], which uses a pronunciation dictionary and a simple rhyme definition to have students find which lines in a text sample rhyme with each other.

Several authors perform a sequence alignment of pairs of phonemic sequences (e.g., Tanasescu, Paget and Inkpen [100]). Sequence alignment is largely used in bioinformatics [113–115] to identify evolutionarily related genomic or protein sequences, but it can also be used for text-based tasks [116], to

identify sequences of letters or words that are more closely related to each other than expected by chance. The score of an alignment between two phoneme sequences is then used to identify whether those sequences rhyme. Hinton and Eastwood [30] use a similar algorithm for the rhymes in Hamilton.

Various extensions to this general model exist. Both Hirjee and Brown [4,5,67] and Bay, Bodily and Ventura [37,38] give more complex ways of scoring these differences between one sequence and another. Bay, Bodily and Ventura score that "kit" rhymes better with "beet" than "kit" rhymes with "boot" because the near-close near-front vowel phoneme /ɪ/ is very similar to the close front vowel phoneme /i/, and not very similar to the close back vowel /u/. These algorithms may allow for missing or added phonemes, such as "fat" rhyming with "past" even though the consonants after the vowel are not identical. Hirjee and Brown's approach uses pre-annotated rhymes (from a rap music lyric corpus) and trains a model to identify alignments between specific phonemes involved in those rhymes. Hirjee and Brown's method builds a rhyme detector pretuned, because of the structure of its training corpus, to African American Vernacular English (AAVE), which many rap artists speak; this pretuning makes it less appropriate for other varieties of English.

Bodily and Ventura use HMM tricks to ensure that generated syllables match each other; this is fairly complex since sampling from a constrained HMM is hard to do mathematically.

Later approaches rely much more on the probabilistic distributions of machine-learning models. Early HMM tricks are used by Dekai Wu's group [6,7,32,107,108], but the first neural-network based rhyming system was the LSTM-based lyric generator GhostWriter by Potash, Romanov and Rumshisky [92,93], which also uses Hirjee and Brown's algorithm [4,5] to verify the rhymes it generates. In general, these approaches tweak the output distributions of a neural network so as to maintain either line-final rhyme or internal rhyme. For example, Li et al. [73] build a transformer-based system that incorporates a mathematical embedding of rhyme information. Miyano and Saito [82] use a generative adversarial

network (GAN) to generate sequences of vowels that have sufficient repetition to match the internal rhyme patterns found in good-quality rap music, then fills in the rest of the words around the vowels; this approach does not guarantee internal coherency. Agarwal and Hamm [34] build acrostics (poems about a theme whose lines all start with the first letters of the word that defines the theme) that attempt to both match the letters of the theme word and rhyme, by tweaking the output probabilities of the model to satisfy both of these constraints. In general, these approaches involve fiddling with the mathematical details of the output distributions of a neural network.

Finally, the "just train the model" approach simply trains a neural network on data with the desired rhyme properties, and either reproduces that distribution (generation) or identifies text that matches the distribution (detection/classification). This approach allows researchers and hobbyists to generate or analyze poetry without building complex models, making the technology accessible to a wider audience.

Language of the rhymes

After we extracted data from the 89 papers under study, we additionally annotated these papers for the language under study (Table 6). We found that 64 papers handled only English rhymes, and another 11 used English and at least one other language (Portuguese, Italian, German, French, and also Bulgarian, Chinese, and Czech, which would not on their own have satisfied our inclusion rules). Additionally, there were thirteen papers not including English rhymes: five papers studying Spanish rhymes, two for each of Italian and Romanian, and one each for Norwegian, Portuguese, French, and Latin. Most of the diversity of language has come in the late period from 2017-2022: only 38 of the 55 papers from that period are about exclusively English rhymes. By contrast, 24 of the 28 middle-period paper were about only English. The early period up to 2005 has only six papers, but they include works focusing on Spanish, Italian, and English plus Bulgarian, as well as an early paper for which we only have the title and cannot give a full description.

Table 3. Primary purpose of algorithms for rhyme detection. Later papers largely concern themselves with generating new rhyming text.

Language	early: 1960s to 2005	mid: 2005 to 2016	late: 2017 to present	Grand Total
English	2	24	38	64
Spanish	1		4	5
English, Portuguese			4	4
Romanian		1	1	2
English, Italian		2		2
Italian	1		1	2
English, French		1		1
French			1	1
English, German, French			1	1
English, Chinese			1	1
unknown	1			1
Norwegian			1	1
Portuguese			1	1
English, Czech, French			1	1
English, Bulgarian	1			1
Latin			1	1
Grand Total	6	28	55	89

The algorithmic complexity seems highest in analysis of English: of the 15 papers that use a complex alignment scoring system, 13 are about English or English plus another language. The only exceptions are rhymes in Latin [83] or Norwegian [51]. By contrast, simple phonemic tricks are used in 6 of the 13 papers that do not include English (versus 19 of the 64 that only use English rhymes), suggesting that authors used simpler algorithms in languages with easier mappings between orthography and pronunciation.

It is clear that the field of rhyme analysis is increasing in linguistic diversity, perhaps in part applying techniques from English to other languages; similarly, authors are taking advantage of the easier task of finding rhymes in orthographically simpler languages.

The significance of rap

Most of the papers about rhyme detection or rhyme analysis in musical contexts deal with rap or hip hop music; see Table 4. Rap music, particularly in the 1990s and 2000s, is characterized by complex rhyme patterns, with internal rhyme, imperfect rhyme, polysyllabic rhyme and other wordplay in its lyrics [117]. See Fig 2 for two examples, taken from Hirjee and Brown [4,5].

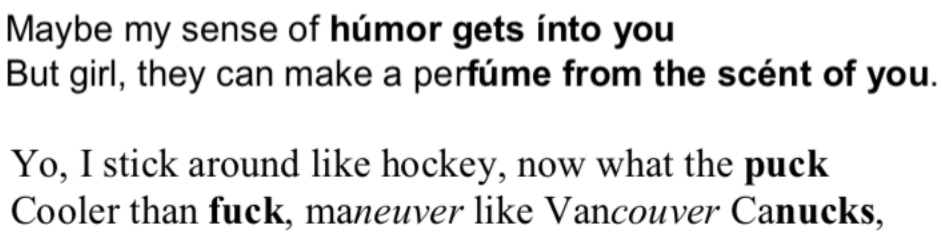

Fig 2. Two examples of complex rhymes, from Hirjee and Brown [4,5]. In the former, from Fabolous's 2001 album *Ghetto Fabolous [118],* a long rhyme pattern moves across multiple words, but has a similar stress pattern (indicated with the accent marks). In the latter, from Pharoahe Monch's 1999 album *Internal Affairs [119]*, multiple rhymes intersect with each other, and are not only found at the ends of lines.

Rhyme complexity has been a prized attribute of good rap music. Authors of systems that analyze rap (like Hirjee and Brown's), or that generate new rap songs (e.g. [6,7]), must be able to cope with this complexity. Lyrics generators, in particular, must be mindful of internal rhymes if they want to be consistent with the customs of the genre.

Most rap music studies have not highlighted the genre's roots in the African American community, though the first study about rhyme detection in rap [4,5,67] and a later very influential one [8] do. Hirjee and Brown highlight the importance of building a custom data set for training their rhyme detector, since rappers may use rhyming pronunciations for words not listed as rhyming in standard pronunciation dictionaries [120], in part because many American rappers use AAVE.

In general, authors of generation systems do not engage with the questions of ethics of artificially generating a genre of music connected to a minority community; generating of rap music is typically treated as interesting primarily because of its linguistic challenge or its cultural importance.

Grey literature

Because poetry, music and language are popular subjects among hobbyists, we also searched a variety of grey literature sources. As described above, we used pre-print servers to identify papers connected to the scholarly process which had not been peer reviewed; source code repositories to identify open-source projects; and Google to identify relevant secondary media.

Our grey literature search yielded several projects of which we were already aware, including Steve Hanov's rhymebrain.com [121]. We also found a number of articles about the phenomenon of computers writing poetry or lyrics (see, for example Faber [122]) ; this topic has of course achieved greater popularity since the release of ChatGPT. There is also a modest space of people selling lyrics generators that highlight the quality of rhymes they create (for example, Chorus [123], or the Rhyme Generator [124]). Several how-to posts appeared, such as Paul Minogue's 2021 article on training Siamese neural networks to identify rhymes [125].

Most of the open-source projects in our search were either projects built in for an undergraduate course, or hobbyist projects; this is common to open-source projects in general. Course projects tended to implement simple phonemic rules, while hobbyist projects often incorporated phoneme-based algorithms, either by copying prior source code or by linking to pre-existing projects. Few of them identify new methods for rhyme detection, and they do not use the more advanced algorithms described in the previous section. It is interesting to us that the academic community and the hobbyist community seem to operate almost independently. Two exceptions are Arlie Coles (a graduate student

at McGill University) who built an LSTM-based rhyme detector that uses advanced machine learning algorithms [126], and Andreas Jörg who built an interface to LLMs for poetry building [127].

The code for Hirjee and Brown's RhymeAnalyzer [4,5] is in SourceForge [128], but since that platform is no longer widely used by open-source developers, and the code for the RhymeAnalyzer has not been migrated to GitHub, there is no evidence that it is widely used by hobbyists. We found two other rhyme-related SourceForge projects; one analyzes Italian rhymes [129], and the other generates heavy metal lyrics [130]. Similarly, there are two relevant PyPI projects; one gives simple phonemic tools [131], while the other is an academic repository of code by Michael Ullyot, from the University of Calgary's English department [132]. We also found four Kaggle projects for poetry generation; one, by Verma et al. [133] implements (essentially) the Hafez algorithm of Ghazvininejad et al. [61]. Others give very little information about how they work.

In Tables 7 and 8, we show the counts of these various grey-lit sources. We believe that searching grey-lit did improve our review: in particular, by highlighting the separation between academic and hobbyist work.

Table 7. Grey literature search results from pre-print servers and open-source repositories. We did not explore all 2260 projects in GitHub found with search for "rhyme"; instead, we limited ourselves to projects that used another term. All searches were done on 30 January 2023.

Database name	Search term	# items retrieved	# items screened	# items added to Zotero for 'full text' review
arXiv https://arxiv.org/search/advanced	title or abstract: rhyme OR lyric OR poem OR poetry OR limerick OR sonnet OR couplet OR stanza OR	418	20	14

	quatrain OR ballad			
Humanities CORE https://hcommons.org/deposits/	computational poetry	7	0	0
	Poetry algorithm	0	0	0
	Rhyming algorithm	0	0	0
GitHub https://github.com/	rhyme generation	17	8	4
	generate rhyme	87	9	5
	detection rhyme	34	6	4
	rhyme detect	16	10	1
	identify rhymes	5	3	3
	rhyme	2260	*	*
Kaggle https://www.kaggle.com/	rhyme OR rhyming OR rhymes	91	8	4
Pypi https://pypi.org/	rhyme	76	8	3
SourceForge https://sourceforge.net/	rhyme	8	2	2
	rhyming	5	1	1

Table 8. Results of Google searches, performed on 30 January 2023, on relevant terms connected to our mapping review. This process yielded newspaper and other articles, as well as several company homepages.

Search terms	**# items retrieved**	**# items screened**	**# items added to Zotero for 'full text' review**
rhyming AND (algorithms OR detect OR generation)	Algorithm: 50 Detect: 50 Generation: 80	Algorithm: 21 Detect: 4 Generation:12	Algorithm: 12 Detect: 2 Generation: 11
rhyming AND artificial intelligence	50	8	7
lyrics AND (algorithms OR detect OR generation)	Algorithm: 10 Detect:10 Generation: 10	Algorithm: 0 Detect: 0 Generation: 1	0

lyrics AND (artificial intelligence OR machine learning OR deep learning OR computational)	10	0	0
poems AND (algorithms OR detect OR generation)	Algorithm: 10 Detect:10 Generation: 40	Algorithm: 8 Detect: 0 Generation: 7	Algorithm: 0 Detect: 0 Generation: 3
poems AND (artificial intelligence)	30	1	1
rap AND (algorithms OR detect OR generation)	Algorithm: 40 Detect: 10 Generation: 20	Algorithm: 4 Detect: 0 Generation: 3	Algorithm: 1 Detect: 0 Generation: 0
rap AND (artificial intelligence)	20	2	0

Observations on the methodology

Sources of included papers

Almost 2% of the database results met the eligibility criteria and the majority of included results came from Scopus. ACL, ACM, and IEEE were the next most valuable, although there was a good deal of overlap between Scopus and the other databases. This overlap is illustrated in Fig 3.

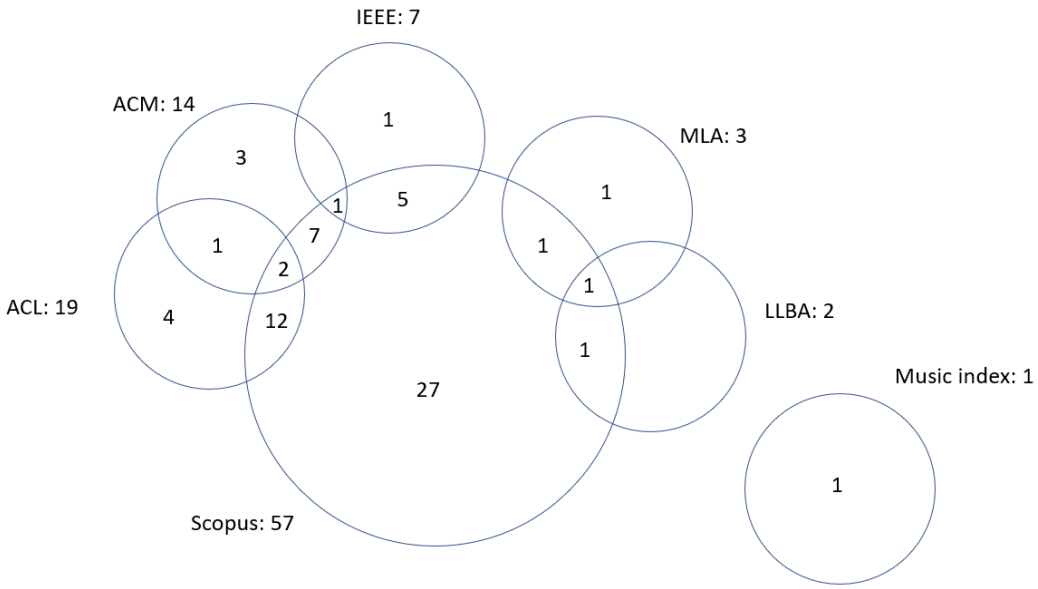

Fig 3. Many included papers were found in more than one database. Scopus found the most unique papers (27). A further 12 were found in Scopus and ACL; 2 were found in Scopus, ACL, and ACM; 7 were found in Scopus and ACM; 1 was found in Scopus, ACM, and IEEE; 5 were found in Scopus and IEEE; 1 was found in Scopus and MLA; 1 was found in Scopus, MLA, and LLBA; 1 was found in Scopus and LLBA. There was minimal overlap between the other databases and no overlap with the Music Index.

About 18% of included articles came from citation searching and about 5.5% were preprints found through the grey literature search. This validates the researchers' concerns that the search may not have been comprehensive and justifies the extra time the reviewers put into screening the included articles' references and papers that cited them.

Database functionality

We believed that the interdisciplinary aspect of this topic necessitated the searching of a range of databases with content from the computer science, digital humanities, and literature disciplines. However, the overlap between the databases indicates that this was not the case. Of the included articles, the Linguistics database didn't return any unique items and the engineering, music, and literature databases only brought in one unique item each. The article, "Prolog realization of a poetry generator" [103] (found only in IEEE) would not have affected the results of the systematic mapping. The paper, "Using Automated Rhyme Detection to Characterize Rhyming Style in Rap Music" [5] (found only in the Music index) describes a major initial study on rap music lyrics, but it is part of a set of papers (Hirjee and Brown, 2009, 2010a, 2010b; Hirjee, 2010) which do appear in other databases we used; it also shares an author with the first author of this review. Mildred Drake's thesis "Internal Rhyme in the Poesía desnuda of Juan Ramón Jiménez, with Notes on Tracing Phonemic Patterns by Computer" [1] (MLA) proved to be a very interesting addition. In addition, there were enough possibly relevant results that the time to search and review the items seemed like a reasonable trade-off. While informed preliminary searching can help identify good databases for inclusion, knowing whether or not a database will be useful is only possible after the search.

The computer science and computational linguistics databases proved valuable in supplementing the Scopus search, but the limits of the ACL anthology brings into question how comprehensive systematic searches can be when repositories of important items do not have advanced search functionality. ACL uses a custom Google search which does not recognize complex Boolean search strings, does not have a field code search, and will find a different number of results for the same search moments later. The researchers were able to take advantage of ACL's full anthology download feature and could run a Boolean search on that file, but review teams without the necessary programming skills would likely have to do without the relevant ACL results. Some of these concerns were also seen with the arXiv repository. A search history that allows for combining previous searches and the option for bulk exporting of citations without an API would make arXiv much more conducive to systematic searching. We echo Gusenbauer and Haddaway's call for database vendors (and repository administrators) to take systematic search functionality seriously and implement or improve upon these necessary criteria in their search systems [17]

This becomes all the more important when considering recent changes in computer science publishing. Computer scientists increasingly upload their papers to arXiv.org and most of these are the preprint version of the paper – before changes from peer review [134]. Therefore, searches on computer science topics cannot rely on traditional research databases alone. And, for this topic in particular, grey literature was noted as a significant source of information because of the amount of relevant work that happens outside of the academy. Based on years of observation, the algorithmic rhyme detection and analysis expert on the review project posited that because of the thin line between research on and application of algorithms and computer programming, that important work may be happening by hobbyists and students that would add value to the review. Therefore we determined that limiting to peer reviewed research literature did not make sense and the grey literature search should include code repositories. It may not be possible for repositories like arXiv to build in the advanced search features

we have come to rely on in traditional databases, but the lack of these features challenge researchers' ability to create truly representative reviews on the application of computer science.

Limitations

We have sought to be comprehensive, but several of our choices may have prevented us from achieving that goal. First, while we discovered several relevant papers via snowball sampling, we chose not to include papers focused on rhyme generation that did not describe a rhyme detection algorithm. This is consistent with our study's goals, but papers that cited the ones we excluded might have themselves included novel rhyme detection algorithms. Second, our choice to exclude rhyme algorithms in languages other than Germanic or Italic ones may mis-shape our understanding of the field. For example, there is a thriving community of researchers generating poetry in classical Chinese style; additionally, there is a strong creativity research community in Helsinki, Finland, but Finnish does not satisfy our requirements. We excluded these languages (and others, such as Arabic, Russian, or Basque) because of our lack of expertise in them, but it is possible that important rhyme algorithms have been developed for these languages.

Similarly, our rule that all papers needed to be written in English may have over-emphasized the role of English in the history of rhyme detection algorithms: it might well be that there are papers written in other languages, including non-English Germanic and Italic languages, which fully develop algorithms for rhyme detection in those languages. In particular, it is hard to be certain that there are not PhD theses like Dr. Mildred Drake's thesis [1], just waiting for us to have found them with the same serendipity we enjoyed in finding hers.

Our challenges in using the broad collection of anthologies, databases and repositories suggest that we may have excluded important papers, either by not including still more collections or by not searching some of them with full success.

Finally, our refusal to consider papers based only on detecting rhyme in audio recordings may have prevented our learning about many interesting definitions of rhyme. Focusing on the sound spoken language might have enabled us to look at how individual performers engage in wordplay around rhyme. Instead, we looked at papers studying languages and dialects with large corpora of written data to train on.

Conclusion

Our systematic review of literature about rhyme has shown us how computer analysis of text has changed across the decades from 1973 to 2022, and in particular how computational generation and analysis of poetry and music lyrics has changed during that period. Early authors in this space focused on simple algorithms for rhyme detection, using orthographic rules or existing rhyme dictionaries for English. Starting in the early 2000s, researchers developed more complex algorithms, focusing on methods like sequence alignment. More recently, there has been a flourishing in this field related to the rise of machine learning algorithms for text analysis and generation. This flourishing includes authors who don't design any algorithm themselves; they simply hand a large data set of rhyming poetry or lyrics to a transformer-based language model. Perhaps the age of algorithms for rhyme detection may be coming to a close.

Acknowledgments

We would like to thank Lauren Byl, Jordan Hale, and Hussein Hirjee for useful conversations at a variety of times.

References

1. Drake MM. Internal Rhyme in the Poesía desnuda of Juan Ramón Jiménez, with notes on tracing phonemic patterns by computer [dissertation]. University of Colorado. 1987. Available: https://login.proxy.lib.uwaterloo.ca/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=cookie,ip,uid&db=mzh&AN=1987080988&site=ehost-live&scope=site

2. Popescu-Belis A, Atrio À, Minder V, Xanthos A, Luthier G, Mattei S, et al. Constrained language models for interactive poem generation. Proceedings of Proceedings of the thirteenth language resources and evaluation conference. Marseille, France: European Language Resources Association; 2022. pp. 3519–3529. Available: https://aclanthology.org/2022.lrec-1.377

3. Gries P, Campbell J, Zingaro D, Fairgrieve T. Poetry form checker. Proceedings of the Western Canadian Conference on Computing Education. New York, New York: ACL; 2019. p. 1. doi:10.1145/3314994.3325080

4. Hirjee H, Brown DG. Rhyme analyzer: An analysis tool for rap lyrics. Proceedings of International Society for Music Information Retrieval Conference: ISMIR 2010. Utrecht, Netherlands: ISMIR; 2010. pp. 1–1.

5. Hirjee H, Brown DG. Using Automated Rhyme Detection to Characterize Rhyming Style in Rap Music. Empir Musicol Rev. 2010;5: 121–145.

6. Wu D, Addanki K, Saers M. Freestyle: A challenge-response system for hip hop lyrics via unsupervised induction of stochastic transduction grammars. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. Lyon, France: ISCA; 2013. pp. 3478–3482. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84906227093&partnerID=40&md5=6c59ad8c4b3102ac49d43beabdc270ab

7. Wu D, Addanki K, Saers M, Beloucif M. Learning to freestyle: Hip hop challenge-response induction via transduction rule segmentation. Proceedings of 2013 Conference on Empirical Methods in Natural Language Processing. Seattle, Washington: ACL; 2013. pp. 102–112. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84926315746&partnerID=40&md5=049f5777d29a5d27ff1a07fbaf49375a

8. Malmi E, Takala P, Toivonen H, Raiko T, Gionis A. DopeLearning: A computational approach to rap lyrics generation. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Online: ACM; 2016. pp. 195–204. doi:10.1145/2939672.2939679

9. Lessard G, Levison M. Computational generation of limericks. Lit Linguist Comput. 2005;20: 89–105. doi:10.1093/llc/fqi019

10. Grant MJ, Booth A. A typology of reviews: An analysis of 14 review types and associated methodologies. Health Inf Libr J. 2009;26: 91–108. doi:https://doi.org/10.1111/j.1471-1842.2009.00848.x

11. Kitchenham B, Brereton P. A systematic review of systematic review process research in software engineering. Inf Softw Technol. 2013;55: 2049–2075. doi:https://doi.org/10.1016/j.infsof.2013.07.010

12. Petersen K, Vakkalanka S, Kuzniarz L. Guidelines for conducting systematic mapping studies in software engineering: An update. Inf Softw Technol. 2015;64: 1–18.

13. Tricco AC, Lillie E, Zarin W, O'Brien KK, Colquhoun H, Levac D, et al. PRISMA extension for scoping reviews (PRISMA-ScR): Checklist and explanation. Ann Intern Med. 2018;169: 467–473. doi:https://doi.org/10.7326/M18-0850

14. Kitchenham B. Procedures for performing systematic reviews. Keele, Staffs (UK): Software Engineering Group Department of Computer Science Keele University; 2004 pp. 1–26. Report No.: TR/SE-0401.

15. Arksey H, O'Malley L. Scoping studies: Towards a methodological framework. Int J Soc Res Methodol. 2005;8: 19–32. doi:10.1080/1364557032000119616

16. Kitchenham B, Charters S. Guidelines for performing systematic literature reviews in software engineering. Keele, Staffs (UK): Software Engineering Group School of Computer Science and Mathematics; 2007 Jul. Report No.: EBSE-2007-01.

17. Gusenbauer M, Haddaway NR. Which academic search systems are suitable for systematic reviews or meta-analyses? Evaluating retrieval qualities of Google Scholar, PubMed, and 26 other resources. Res Synth Methods. 2020;11: 181–217. doi:https://doi.org/10.1002/jrsm.1378

18. Garousi V, Felderer M, Mäntylä MV. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Inf Softw Technol. 2019;106: 101–121.

19. Zhang H, Mao R, Huang H, Dai Q, Zhou X, Shen H, et al. Processes, challenges and recommendations of Gray Literature Review: An experience report. Inf Softw Technol. 2021;137: 106607.

20. Godin K, Stapleton J, Kirkpatrick SI, Hanning RM, Leatherdale ST. Applying systematic review search methods to the grey literature: A case study examining guidelines for school-based breakfast programs in Canada. Syst Rev. 2015;4: 138. doi:10.1186/s13643-015-0125-0

21. Hanson K. Rhyme. Brown K, editor. Encyclopedia of Language & Linguistics. Oxford: Elsevier; 2006. pp. 605–616. doi:10.1016/B0-08-044854-2/00542-3

22. ProQuest. LLBA Thesaurus. In: - Linguistics and Language Behavior Abstracts (LLBA) [Internet]. [cited 4 Jul 2023]. Available: https://www.proquest.com/llba/thesaurus/browse/$N/queryTermField/false/false/false?accountid=14906

23. EBSCOhost. CMMC Subjects. In: Communication & Mass Media Complete Subjects [Internet]. [cited 4 Jul 2023]. Available: https://web.p.ebscohost.com/ehost/thesaurus?vid=2&sid=012e6fa4-0fc8-47bd-a7f8-b6db612e4fb9%40redis

24. Nigrin AL. SONNET: A self-organizing neural network that classifies multiple patterns simultaneously. Proceedings of 1990 IJCNN International Joint Conference on Neural Networks. San Diego, California: IEEE; 1990. pp. 313–318 vol.2. doi:10.1109/IJCNN.1990.137732

25. Merriam-Webster. Definition of RIME. 2022. Available: https://www.merriam-webster.com/dictionary/rime

26. Lamb C, Brown DG, Clarke CLA. A taxonomy of generative poetry techniques. J Math Arts. 2017;11: 159–179. doi:10.1080/17513472.2017.1373561

27. Sawicki P, Grzes M, Jordanous A, Brown D, Peeperkorn M. Training GPT-2 to represent two Romantic-era authors: challenges, evaluations and pitfalls. Proceedings of the 13th International Conference on Computational Creativity. Bozen-Bolzano, Italy: ACC; 2022. pp. 34–43. Available: http://computationalcreativity.net/iccc22/papers/ICCC-2022_paper_45.pdf

28. Sawicki P, Grzes M, Goes F, Brown D, Peeperkorn M, Khatun A. Bits of Grass: Does GPT already know how to write like Whitman? arXiv; 2023. doi:10.48550/arXiv.2305.11064

29. Sawicki P, Grzes M, Goes F, Brown D, Peeperkorn M, Khatun A, et al. On the power of special-purpose GPT models to create and evaluate new poetry in old styles. Proceedings of the Fourteenth International Conference on Computational Creativity. Waterloo, Canada: ACC; 2023. Available: https://kar.kent.ac.uk/101234/

30. Hinton E, Eastwood J. Playing with pop culture: Writing an algorithm to analyze and visualize lyrics from the musical “Hamilton.” Proceedings of the Computation + Journalism Symposium. Stanford, California: Stanford University; 2016.

31. Pinto MP. Using Computers to Analyze Poems: A Phonological Metrification. CTJ J. 1991; 16–24.

32. Addanki K, Wu D. Unsupervised rhyme scheme identification in hip hop lyrics using hidden markov models. In: Dediu A-H, Martín-Vide C, Mitkov R, Truthe B, editors. Proceedings of Statistical Language and Speech Processing. Berlin, Heidelberg: Springer; 2013. pp. 39–50. doi:10.1007/978-3-642-39593-2_3

33. Addanki K, Wu D. Evaluating improvised hip hop lyrics - Challenges and observations. 2014. pp. 4616–4623.

34. Agarwal R, Kann K. Acrostic poem generation. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Online: EMNLP; 2020. pp. 1230–1240. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85102166039&partnerID=40&md5=2ea456e8c12f3a628f1cd2744b88b7bf

35. Anand A, Anand A, Maiyuran J, Shum M. Rap lyric generation: A phoneme-based LSTM approach. github [Preprint]; 2017. Available: https://michaelshum.github.io/files/rap-lyric-generation.pdf

36. Barbieri G, Pachet F, Roy P, Esposti MD. Markov constraints for generating lyrics with style. Front Artif Intell Appl. 2012;242: 115–120. doi:10.3233/978-1-61499-098-7-115

37. Bay B, Bodily P, Ventura D. Text transformation via constraints and word embedding. In: Goel A., Jordanous A., Pease A., editors. Proceedings of the 8th International Conference on Computational Creativity, ICCC 2017. Atlanta, Georgia: ACC; 2017. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85126501444&partnerID=40&md5=8195791f4df2c0f5da370de2e48d8cb6

38. Bay B, Bodily P, Ventura D. Dynamically scoring rhymes with phonetic features and sequence alignment. ICTAI 2019: Proceedings of the International Conference on Tools with Artificial Intelligence. Portland: IEEE; 2019. pp. 1581–1585. doi:10.1109/ICTAI.2019.00228

39. Belouadi J, Eger S. ByGPT5: End-to-end style-conditioned poetry generation with token-free language models. arXiv:2212.10474 [preprint]; 2022. doi:10.48550/arXiv.2212.10474

40. Benhardt J, Hase P, Zhu L, Rudin C. Shall I compare thee to a machine-written sonnet? An approach to algorithmic sonnet generation. arXiv; 2019. doi:10.48550/arXiv.1811.05067

41. Bianco PD, Mindlin I, Lanzarini L, Ronchetti F, Hasperué W, Quiroga F. Structured text generation for Spanish freestyle battles using neural networks. Proceedings of 2021 XLVII latin american computing conference (CLEI). Cartago, Costa Rica: IEEE; 2021. pp. 1–9. doi:10.1109/CLEI53233.2021.9639929

42. Bodily P, Ventura D. Steerable music generation which satisfies long-range dependency constraints. Trans Int Soc Music Inf Retr. 2022;5: 71–86. doi:10.5334/tismir.97

43. Bulger D, Powles T. Constrained sampling of Markov chains. Proceedings of 22nd International Congress on Modelling and Simulation, MODSIM 2017. Hobart, Tasmania: MSSANZ; 2017. pp. 466–472. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85080933969&partnerID=40&md5=a80803906f1606a28b978099cd79816d

44. Byrd RJ, Chodorow MS. Using an on-line dictionary to find rhyming words and pronunciations for unknown words. Proceedings of the 23rd annual meeting on Association for Computational Linguistics. Chicago, Illinois: ACL; 1985. pp. 277–283. doi:10.3115/981210.981244

45. Campos F. A conversational chatbot that can rhyme [thesis]. California State University. 2022. Available: https://scholarworks.csun.edu/bitstream/handle/10211.3/223036/Campos-Froilan-thesis-2022.pdf?sequence=1

46. Chang J-W, Hung JC, Lin K-C. Singability-enhanced lyric generator with music style transfer. Comput Commun. 2021;168: 33–53. doi:10.1016/j.comcom.2021.01.002

47. Ciobanu A, Dinu LP. On the Romanian rhyme detection. Proceedings of COLING 2012: Demonstration papers. Mumbai, India: COLING; 2012. pp. 87–94. Available: https://aclanthology.org/C12-3011

48. Colton S, Goodwin J, Veale T. Full-FACE poetry generation. Proceedings of the 3rd International Conference on Computational Creativity, ICCC 2012. Dublin, Ireland: ACC; 2012. pp. 95–102. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84984853573&partnerID=40&md5=a4bc6ee9c415e67199104d036cc1216a

49. Delmonte R. A computational approach to poetic structure, rhythm and rhyme. Proceedings of the First Italian Conference on Computational Linguistics CLiC-it 2014 & and of the Fourth International Workshop EVALITA 2014. Pisa, Italy: Pisa University Press; 2014. Available: http://digital.casalini.it/9788867414727

50. El Bolock A. Automatic poetry generation using CHR [thesis]. German University in Cairo. 2014. Available: http://www.informatik.uni-ulm.de/pm/fileadmin/pm/home/fruehwirth/drafts/bolock-master-thesis-draft.pdf

51. Enstad TR. Norwegian poetry generation and rhyme modelling [thesis]. University of Oslo. 2022. Available: https://www.duo.uio.no/handle/10852/95620

52. Fang M, Jiang X, Zhao Q, Jiang Y. Automatic choosing of English rhymes in translation of Chinese ancient poems. Proceedings of 2009 2nd International Symposium on Knowledge Acquisition and Modeling, KAM 2009. Wuhan, China: IEEE; 2009. pp. 434–437. doi:10.1109/KAM.2009.78

53. Fernandez ACT, Tarnate KJM, Devaraj M. Deep rapping: Character level neural models for automated rap lyrics composition. Int J Innov Technol Explor Eng. 2018;8: 306–311.

54. Ferraz de Arruda H, Reia SM, Silva FN, Amancio DR, da Fontoura Costa L. Finding contrasting patterns in rhythmic properties between prose and poetry. Phys Stat Mech Its Appl. 2022;598. doi:10.1016/j.physa.2022.127387

55. Fusu E-C, Pura M-L. Roetry: First steps towards computer generated poetry in romanian language. Proceedings of the 12th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2020. Romania: IEEE; 2020. doi:10.1109/ECAI50035.2020.9223175

56. Gatti L, Özbal G, Stock O, Strapparava C. Automatic generation of lyrics parodies. MM 2017 - Proceedings of the 2017 ACM Multimedia Conference. Mountain View, California: ACM; 2017. pp. 485–491. doi:10.1145/3123266.3123410

57. Gatti L, Ozbal G, Stock O, Strapparava C. To sing like a mockingbird. Valencia, Spain: ACL; 2017. pp. 298–304. doi:10.18653/v1/e17-2048

58. Genzel D, Uszkoreit J, Och F. “Poetic” statistical machine translation: Rhyme and meter. Proceedings of EMNLP 2010 - Conference on Empirical Methods in Natural Language Processing. Cambridge Massachusetts: ACL; 2010. pp. 158–166. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-80053281400&partnerID=40&md5=007e4c92865af1b3284de694f23f0791

59. Gervás P. Template-free construction of rhyming poems with thematic cohesion. Proceedings of the workshop on computational creativity in natural language generation (CC-NLG 2017). Santiago de Compostela, Spain: ACL; 2017. pp. 21–28. doi:10.18653/v1/W17-3903

60. Ghazvininejad M, Shi X, Choi Y, Knight K. Generating topical poetry. Proceedings of EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing. Austin, Texas: ACL; 2016. pp. 1183–1191. doi:10.18653/v1/d16-1126

61. Ghazvininejad M, Shi X, Priyadarshi J, Knight K. Hafez: An interactive poetry generation system. Proceedings of ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics. Vancouver, Canada: ACL; 2017. pp. 43–48. doi:10.18653/v1/P17-4008

62. Greene E, Bodrumlu T, Knight K. Automatic analysis of rhythmic poetry with applications to generation and translation. Proceedings of EMNLP 2010 - Conference on Empirical Methods in

Natural Language Processing. Cambridge, Massachusetts: ACL; 2010. pp. 524–533. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-80053256340&partnerID=40&md5=9905121162b868e46eab352b9ac6adb2

63. Haider T, Kuhn J. Supervised rhyme detection with siamese recurrent networks. Santa Fe, New Mexico: ACL; 2018. pp. 81–86. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85091571021&partnerID=40&md5=e9e0ff7db160696f1f6933430e732a2b

64. Hegade P, Joshi RM, Hegde VG, Kale T, Basavaraddi S. Po-miner: a web mining poem generator and its security model. SN Comput Sci. 2021;2. doi:10.1007/s42979-021-00802-6

65. Hernández-Lorenzo L, Diaz A, Perez A, Ros S, González-Blanco E. Exploring Spanish contemporary song lyrics through Digital Humanities methods: Some thematic and structural properties. Digit Scholarsh Humanit. 2022;37: 738–746. doi:10.1093/llc/fqab083

66. Hernadez M. Automatic Generation of Hip-Hop and Rap Lyrics [dissertation]. Université Sorbonne Nouvelle. 2018. Available: http://www.tal.univ-paris3.fr/plurital/memoires/HERNANDEZ_madeleine-RD-1718.pdf

67. Hirjee H, Brown DG. Automatic detection of internal and imperfect rhymes in rap lyrics. Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009. Kobe, Japan: ISMIR; 2009. pp. 711–716. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84873576245&partnerID=40&md5=1a37da783c8ce33e0d11e4d969e850ef

68. Hirjee H. Rhyme, rhythm, and rhubarb: Using probabilistic methods to analyze hip hop, poetry, and misheard lyrics [thesis]. University of Waterloo. 2010.

69. Hopkins J, Kiela D. Automatically generating rhythmic verse with neural networks. Vancouver, Canada: ACL; 2017. pp. 168–178. doi:10.18653/v1/P17-1016

70. Jensen CZ, Sørhaug E. The Perfect Rap Lyrics [thesis]. Master thesis, NTNU. 2021. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2834978

71. Jhamtani H, Mehta SV, Carbonell J, Berg-Kirkpatrick T. Learning rhyming constraints using structured adversaries. Hong Kong, China: ACL; 2019. pp. 6025–6031. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85081894649&partnerID=40&md5=8ed2a0cd2a9d76027fb51009af7f6c5a

72. Lau JH, Cohn T, Baldwin T, Brooke J, Hammond A. Deep-speare: A joint neural model of poetic language, meter and rhyme. Melbourne, Australia: ACL; 2018. pp. 1948–1958. doi:10.18653/v1/p18-1181

73. Li P, Zhang H, Liu X, Shi S. Rigid formats controlled text generation. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Online: ACL; 2020. pp. 742–751. doi:10.18653/v1/2020.acl-main.68

74. Liang H, Li Q, Wang H, Li H, Wang J, Sun Z, et al. HAVAE: Learning prosodic-enhanced representations of rap lyrics. Proceedings of PRICAI 2018: Trends in Artificial Intelligence PRICAI 2018 Lecture Notes in Computer Science, vol 11012. Nanjing, China: Springer; 2018. pp. 1–15.

75. Liang H, Wang H, Li Q, Wang J, Xu G, Chen J, et al. A general framework for learning prosodic-enhanced representation of rap lyrics. World Wide Web. 2019;22: 2267–2289. doi:10.1007/s11280-019-00672-2

76. Lo K-L, Ariss R, Kurz P. GPoeT-2: A GPT-2 Based Poem Generator. arXiv 2205.08847 [Preprint]; 2022. doi:10.48550/arXiv.2205.08847

77. Lu J, Eirinaki M. Can a machine win a Grammy? An evaluation of AI-generated song lyrics. Proceedings of 2021 IEEE International Conference on Big Data, Big Data 2021. Online: IEEE; 2021. pp. 4896–4905. doi:10.1109/BigData52589.2021.9671431

78. Manjavacas E, Karsdorp F, Kestemont M. Generation of hip-hop lyrics with hierarchical modeling and conditional templates. Proceedings of INLG 2019 - 12th International Conference on Natural Language Generation. Tokyo, Japan: INLG; 2019. pp. 301–310. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85087174596&partnerID=40&md5=5446ea35766c209aeee3fbc39c00f2bb

79. Medina-Urrea A, Torres-Moreno J-M. RIMAX: Ranking Semantic Rhymes by calculating Definition Similarity. arXiv 1912.09558 [Preprint]; 2019. doi:10.48550/arXiv.1912.09558

80. Melvill-Smith S, Krause AE, North AC. Song popularity and processing fluency of lyrics. Psychol Music. 2022. doi:10.1177/03057356221118400

81. Mittmann A, von Wangenheim A, dos Santos AL. Aoidos: A system for the automatic scansion of poetry written in Portuguese. In: Gelbukh A., editor. Proceedings of Computational Linguistics and Intelligent Text Processing CICLing 2016 Lecture Notes in Computer Science, vol 9624. Konya, Turkey: Springer; 2018. pp. 611–628. doi:10.1007/978-3-319-75487-1_46

82. Miyano T, Saito H. Rap Lyrics Generation Using Vowel GAN. Commun Comput Inf Sci. 2020;1215 CCIS: 307–318. doi:10.1007/978-981-15-6168-9_26

83. Nagy B. Rhyme in classical Latin poetry: Stylistic or stochastic? Digit Scholarsh Humanit. 2022;37: 1097–1118.

84. Nikolov NI, Malmi E, Northcutt CG, Parisi L. Rapformer: Conditional rap lyrics generation with denoising autoencoders. Proceedings of INLG 2020 - 13th International Conference on Natural Language Generation. Dublin, Ireland: ACL; 2020. pp. 360–373. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85108730448&partnerID=40&md5=c58f7b6ef00c65dafc046e17c6750a4c

85. Oliveira HG. A REST service for poetry generation. Proceedings of 6th Symposium on Languages, Applications and Technologies, SLATE 2017. Vila do Conde, Portugal: Schloss Dagstuhl – Leibniz-Zentrum für Informatik; 2017. doi:10.4230/OASIcs.SLATE.2017.12

86. Oliveira HG, Mendes T, Boavida A. Co-PoeTryMe: A co-creative interface for the composition of poetry. Proceedings of INLG 2017 - 10th International Natural Language Generation Conference. Santiago de Compostela, Spain: ACL; 2017. pp. 70–71. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85069513269&partnerID=40&md5=7c104cf657f64e2a775b70987761655c

87. Oliveira HG, Hervás R, Díaz A, Gervás P. Multilingual extension and evaluation of a poetry generator. Nat Lang Eng. 2017;23: 929–967. doi:10.1017/S1351324917000171

88. Oliveira HG, Mendes T, Boavida A, Nakamura A, Ackerman M. Co-PoeTryMe: Interactive poetry generation. Cogn Syst Res. 2019;54: 199–216. doi:10.1016/j.cogsys.2018.11.012

89. Peterson C. Generating rhyming poetry using LSTM recurrent neural networks [thesis]. Thesis. 2019. Available: https://dspace.library.uvic.ca/handle/1828/10801

90. Plamondon MR. Virtual verse analysis: Analysing patterns in poetry. Lit Linguist Comput. 2006;21: 127–141. doi:10.1093/llc/fql011

91. Plecháč P. A collocation-driven method of discovering rhymes (in Czech, English, and French poetry). In: Fidler M, Cvrček V, editors. Taming the corpus: From inflection and lexis to interpretation. Cham: Springer International Publishing; 2018. pp. 79–95. doi:10.1007/978-3-319-98017-1_5

92. Potash P, Romanov A, Rumshisky A. Ghostwriter: Using an lstm for automatic rap lyric generation. Proceedings of EMNLP 2015: Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: ACL; 2015. pp. 1919–1924. doi:10.18653/v1/d15-1221

93. Potash P, Romanov A, Rumshisky A. Evaluating creative language generation: The case of rap lyric ghostwriting. arXiv 1612.03205 [Preprint]; 2016. Available: https://arxiv.org/abs/1612.03205

94. Ram N, Gummadi T, Bhethanabotla R, Savery RJ, Weinberg G. Say what? Collaborative pop lyric generation using multitask transfer learning. Proceedings of the 9th International Conference on Human-Agent Interaction. New York, New York: ACL; 2021. pp. 165–173. doi:10.1145/3472307.3484175

95. Ratawal Y, Makhloga VS, Raheja K, Chadha P, Bhatt N. PoemAI: Text Generator Assistant for Writers. Lect Notes Netw Syst. 2022;213: 575–584. doi:10.1007/978-981-16-2422-3_45

96. Reddy S, Knight K. Unsupervised discovery of rhyme schemes. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland, Oregon: ACL; 2011. pp. 77–82. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84859025830&partnerID=40&md5=99c07721a7ef7aa6b7426677be5d34cc

97. Robey D. Rhymes in the renaissance epic: A computer analysis of pulci, boiardo, ariosto and tasso. Roman Stud. 1991;9: 97–111. doi:10.1179/ros.1991.9.2.97

98. Savery R, Zahray L, Weinberg G. Shimon the Rapper: A Real-Time System for Human-Robot Interactive Rap Battles. arXiv; 2020. doi:10.48550/arXiv.2009.09234

99. Singhi A, Brown DG. Are poetry and lyrics all that different? In: Wang H.-M., Yang Y.-H., Lee J.H., editors. Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014. Taipei, Taiwan: ISMIR; 2014. pp. 471–476. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85015318084&partnerID=40&md5=1a08f3f257826cea72ba7ab74f102866

100. Tanasescu C, Paget B, Inkpen D. Automatic classification of poetry by meter and rhyme. Proceedings of the 29th International Florida Artificial Intelligence Research Society Conference, FLAIRS 2016. Key Largo, Florida: AAAI; 2016. pp. 244–249. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85004093075&partnerID=40&md5=f4e90d21fc53525bf66ad8e85d572cd0

101. Tian Y, Peng N. Zero-shot sonnet generation with discourse-level planning and aesthetics features. Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Seattle, United States: ACL; 2022. pp. 3587–3597. doi:10.18653/v1/2022.naacl-main.262

102. Uthus D, Voitovich M, Mical RJ. Augmenting Poetry Composition with Verse by Verse. 2022. doi:10.48550/arXiv.2103.17205

103. Atanassova V, Pulov S. Prolog realization of a poetry generator. Proceedings of First International IEEE Symposium Intelligent Systems. Varna, Bulgaria: IEEE; 2002. pp. 52–53. doi:10.1109/IS.2002.1042586

104. Wöckener J, Haider T, Miller T, Nguyen TTL, Nguyen T-K, Pham MV, et al. End-to-end style-conditioned poetry generation: What does it take to learn from examples alone? Proceedings of 5th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature, LaTeCHCLfL 2021 - Co-located with the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021. Punta Cana, Dominican Republic (online): ACL; 2021. pp. 57–66. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85138463950&partnerID=40&md5=9e2a5e482bf2204cfe71c9a89a14796c

105. Wang J, Zhang X, Zhou Y, Suh C, Rudin C. There once was a really bad poet, it was automated but you didn’t know it. Trans Assoc Comput Linguist. 2021;9: 605–620. doi:10.1162/tacl_a_00387

106. Wechsler LR, Sondak NE. The simulation of a creative process—the composition of man-machine poetry. Proceedings of the ACM Annual Conference. New York, New York: ACM; 1973. p. 434.5-435. doi:10.1145/800192.805774

107. Wu D, Addanki K. Learning to rap battle with bilingual recursive neural networks. Proceedings of 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013). Beijing, China: IJCAI; 2015. pp. 2524–2530. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84949772965&partnerID=40&md5=71a9b6aabfc793898283916cf9e0ff68

108. Wu D, Addanki K. Neural Versus symbolic rap battle bots. Proceedings of 41st International Computer Music Conference, ICMC 2015. Denton, Texas: ICMA; 2015. pp. 210–213.

109. Zugarini A, Melacci S, Maggini M. Neural Poetry: Learning to Generate Poems Using Syllables. Lect Notes Comput Sci Subser Lect Notes Artif Intell Lect Notes Bioinforma. 2019;11730 LNCS: 313–325. doi:10.1007/978-3-030-30490-4_26

110. Zugarini A, Pasqualini L, Melacci S, Maggini M. Generate and revise: Reinforcement learning in neural poetry. 2021 International Joint Conference on Neural Networks (IJCNN). Online: IEEE; 2021. doi:10.1109/IJCNN52387.2021.9533573

111. Drake MM. [Review of Poesía Hispanoamericana]. Hispania. 1994;77: 870–870. doi:10.2307/345743

112. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention Is All You Need. arXiv; 2017. doi:10.48550/arXiv.1706.03762

113. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol. 1970;48: 443–453. doi:10.1016/0022-2836(70)90057-4

114. Smith TF, Waterman MS. Identification of common molecular subsequences. J Mol Biol. 1981;147: 195–197. doi:10.1016/0022-2836(81)90087-5

115. Durbin R, Eddy SR, Krogh A, Mitchison G. Biological sequence analysis: Probabilistic models of proteins and nucleic acids. Cambridge: Cambridge University Press; 1998. doi:10.1017/CBO9780511790492

116. Levenshtein VI. Binary codes capable of correcting deletions, insertions, and reversals. Sov Phys Dokl. 1966;10: 707–710.

117. Alim HS. On some serious next millennium rap ishhh: Pharoahe Monch, hip hop poetics, and the internal rhymes of internal affairs. J Engl Linguist. 2003;31: 60–84. doi:10.1177/0075424202250619

118. Fabolous. Ghetto Fabolous. Elektra Records; 2001.

119. Pharoahe Monch P. Internal Affairs. Rawkus Records; 1999.

120. Lenzo K. The CMU Pronouncing Dictionary. In: The CMU Pronouncing Dictionary [Internet]. [cited 8 Jun 2023]. Available: http://www.speech.cs.cmu.edu/cgi-bin/cmudict

121. Hanov S. A rhyming engine. In: Steve Hanov's Blog [Internet]. 2008 [cited 30 Jan 2023]. Available: http://stevehanov.ca/blog/?id=8

122. Faber T. Rapper's delight or weapons-grade nonsense? The app that uses AI to help MCs bust a rhyme. In: The Guardian [Internet]. 2022 [cited 30 Jan 2023]. Available: https://www.theguardian.com/music/2022/aug/23/ai-rap-technology-music

123. Hicks W. Boost your songwriting process with the power of AI. In: Chorus [Internet]. 6 Dec 2022 [cited 30 Jan 2023]. Available: https://writewithchorus.com/boost-your-songwriting-process-with-chorus

124. Iulian F. Rhyme Generator. In: APK for Android Download [Internet]. 2022 [cited 30 Jan 2023]. Available: https://m.apkpure.com/rhyme-generator/rhyme.generator.flst

125. Minogue P. Using Siamese neural networks to create a simple rhyme detection system. In: Paul Minogue [Internet]. 14 Feb 2021 [cited 7 Jun 2023]. Available: https://paulminogue.com/posts/eca77f81-ff5f-4bf5-b8a6-16a50558c81a

126. Coles A. deep-rhyme-detection [software]. 2023. Available: https://github.com/a-coles/deep-rhyme-detection

127. Jörg A. Poetry Generation with large language models like GPT2 or GPT3 [software]. 2023. Available: https://github.com/andjoer/llm_poetry_generation

128. Meltedsnowball, Necropolitan. Rhyme Analyzer [software]. 2013. Available: https://sourceforge.net/projects/rhymeanalyzer/

129. Baliello. Rhyme - Rimario [software]. 2014. Available: https://sourceforge.net/projects/rimario/

130. Wansti. Battlecry [software]. 2016. Available: https://sourceforge.net/projects/battlecry/

131. Danis N. phonk: Some helpful phonetics and phonology functions. 2021. Available: https://github.com/nickdanis/phonk

132. Harkema J. Augmented-Criticism-Lab-Toolkit: A set of tools for connecting to the Augmented Criticism Lab. 2018. Available: https://git.joshharkema.com/jharkema/augemented-criticism-lab-tools

133. Verma T. poem_gen. 2020. Available: https://kaggle.com/code/tathagatv/poem-gen

134. Sutton C, Gong L. Popularity of arXiv.org within Computer Science. arXiv; 2017. doi:10.48550/arXiv.1710.05225

S1 Appendix

Table 9. ACM Digital Library Expanded Guide to the Computing Literature search, performed November 21, 2022.

Search	Query	Field	Results
1	troch* OR assonan* OR consonan* OR "scansion" OR "scansions" OR iamb* OR sonnet* OR limerick* OR prosod* OR caden* OR freestyl* OR syllab*	Title	3,859
2	troch* OR assonan* OR consonan* OR "scansion" OR "scansions" OR iamb* OR sonnet* OR limerick* OR prosod* OR caden* OR freestyl* OR syllab*	Abstract	19,113
3	troch* OR assonan* OR consonan* OR "scansion" OR "scansions" OR iamb* OR sonnet* OR limerick* OR prosod* OR caden* OR freestyl* OR syllab*	Author Keyword	886
4	1 OR 2 OR 3		20,086
5	music* OR song* OR poem* OR poet* OR lyric* OR verse*	Title	9,641
6	music* OR song* OR poem* OR poet* OR lyric* OR verse*	Abstract	22,889
7	music* OR song* OR poem* OR poet* OR lyric* OR verse*	Author Keyword	5,763
8	5 OR 6 OR 7		24,965
9	4 AND 8		356
10	rhyme* OR rhyming OR quatrain* OR couplet*	Title	40
11	rhyme* OR rhyming OR quatrain* OR couplet*	Abstract	184
12	rhyme* OR rhyming OR quatrain* OR couplet*	Author Keyword	19
13	10 OR 11 OR 12		192
14	9 OR 13		533

Table 10. IEEE Xplore Digital Library, performed November 21, 2022.

Search	Query	Field	Results
1	couplet OR stanza OR ballad OR troch* OR assonan* OR consonan* OR caden* OR scansion OR iamb* OR prosod* OR freestyl* OR rhythmical OR rhythmically OR rhythmically OR syllable OR syllabify OR syllabification	All Metadata	11,688
2	music OR musical OR musician OR song OR poem OR poet* OR lyric OR lyrical OR verse	All Metadata	80,651
3	1 AND 2	All Metadata	329

4	rhyme OR rhyming	All Metadata	199
5	3 OR 4		519

Table 11. Music Index via EBSCOhost, performed November 21, 2022.

Search	Query	Field	Results
1	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	TI	6,665
2	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	AB	21,437
3	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	KW	2,439
4	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	SU	29,524
5	1 OR 2 OR 3 OR 4		41,567
6	rhyme* OR rhyming OR quatrain* OR couplet* OR troch* OR limerick* OR sonnet* OR consonan* OR assonan* OR scansion* OR caden* OR iamb* OR prosod* OR freestyl* OR villanelle* OR tercet* OR cinquain* OR sestet* OR ballad*	TI	5,582
7	rhyme* OR rhyming OR quatrain* OR couplet* OR troch* OR limerick* OR sonnet* OR consonan* OR assonan* OR scansion* OR caden* OR iamb* OR prosod* OR freestyl* OR villanelle* OR tercet* OR cinquain* OR sestet* OR ballad*	AB	6,412
8	rhyme* OR rhyming OR quatrain* OR couplet* OR troch* OR limerick* OR sonnet* OR consonan* OR assonan* OR scansion* OR	KW	1,215

	caden* OR iamb* OR prosod* OR freestyl* OR villanelle* OR tercet* OR cinquain* OR sestet* OR ballad*		
9	rhyme* OR rhyming OR quatrain* OR couplet* OR troch* OR limerick* OR sonnet* OR consonan* OR assonan* OR scansion* OR caden* OR iamb* OR prosod* OR freestyl* OR villanelle* OR tercet* OR cinquain* OR sestet* OR ballad*	SU	5,485
10	6 OR 7 OR 8 OR 9		12,709
11	5 AND 10		218
12	Limit 11 to English		216

Table 12. MLA International Bibliography via EBSCOhost, performed November 21, 2022.

Search	Query	Field	Results
1	troch* OR assonan* OR scansion* OR iamb* OR consonan* OR caden* OR prosod* OR rhythm* OR meter* OR metre* OR metric*	TI	10,963
2	troch* OR assonan* OR scansion* OR iamb* OR consonan* OR caden* OR prosod* OR rhythm* OR meter* OR metre* OR metric*	AB	3,219
3	troch* OR assonan* OR scansion* OR iamb* OR consonan* OR caden* OR prosod* OR rhythm* OR meter* OR metre* OR metric*	SU	25,752
4	1 OR 2 OR 3		32,168
5	music* OR song* OR poem* OR poet* OR lyric* OR verse* OR tune*	TI	150,846
6	music* OR song* OR poem* OR poet* OR lyric* OR verse* OR tune*	AB	24,373
7	music* OR song* OR poem* OR poet* OR lyric* OR verse* OR tune*	SU	399,507
8	5 OR 6 OR 7		454,382
9	4 AND 8		6,766
10	rhyme* OR rhyming OR quatrain* OR limerick* OR sonnet* OR couplet* OR villanelle* OR ballad* OR stanza*	TI	9,096
11	rhyme* OR rhyming OR quatrain* OR limerick* OR sonnet* OR couplet* OR villanelle* OR ballad* OR stanza*	AB	1,950
12	rhyme* OR rhyming OR quatrain* OR limerick* OR sonnet* OR couplet* OR villanelle* OR ballad* OR stanza*	SU	13,154
13	10 OR 11 OR 12		17,851
14	9 OR 13		23,866
15	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*"	TI	10,490

	OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"		
16	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	AB	3,470
17	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	SU	25,646
18	15 OR 16 OR 17		31,182
19	14 AND 18		143
20	Limit 19 to English		121

Table 13. Communication & Mass Media Complete via EBSCOhost, performed November 21, 2022.

Search	Query	Field	Results
1	troch* OR assonan* OR scansion* OR iamb* OR consonan* OR caden* OR prosod* OR metric*	TI	2,886
2	troch* OR assonan* OR scansion* OR iamb* OR consonan* OR caden* OR prosod* OR metric*	AB	10,827
3	troch* OR assonan* OR scansion* OR iamb* OR consonan* OR caden* OR prosod* OR metric*	KW	2,423
4	troch* OR assonan* OR scansion* OR iamb* OR consonan* OR caden* OR prosod* OR metric*	SU	3,047
5	1 OR 2 OR 3 OR 4		12,076
6	music* OR song* OR poem* OR poet* OR lyric* OR verse*	TI	8,004
7	music* OR song* OR poem* OR poet* OR lyric* OR verse*	AB	20,136
8	music* OR song* OR poem* OR poet* OR lyric* OR verse*	KW	3,934
9	music* OR song* OR poem* OR poet* OR lyric* OR verse*	SU	13,196
10	6 OR 7 OR 8 OR 9		24,427
11	5 AND 10		508
12	rhyme* OR rhyming OR quatrain* OR limerick* OR sonnet* OR couplet*	TI	231
13	rhyme* OR rhyming OR quatrain* OR limerick* OR sonnet* OR couplet*	AB	829

14	rhyme* OR rhyming OR quatrain* OR limerick* OR sonnet* OR couplet*	KW	111
15	rhyme* OR rhyming OR quatrain* OR limerick* OR sonnet* OR couplet*	SU	311
16	DE "RHYME" OR DE "ASSONANCE" OR DE "LIMERICKS" OR DE "STANZAS"	Any field	187
17	12 OR 13 OR 14 OR 15 OR 16		947
18	11 OR 17		1,406
19	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	TI	11,134
20	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	AB	52,333
21	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	KW	9,619
22	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	SU	52,088
23	DE "ARTIFICIAL neural networks" OR DE "MULTILAYER perceptrons" OR DE "DEEP learning" OR DE "ARTIFICIAL neural networks" OR DE "COMPUTATIONAL linguistics methodology" OR DE "COMPUTATIONAL linguistics" OR DE "AUTOMATIC spelling-to-sound conversion" OR DE "LANGUAGE identification (Computational linguistics)" OR DE "NAMED-entity recognition" OR DE "NETWORK grammar" OR DE "SENTIMENT analysis" OR DE "SPEECH processing systems" OR DE "INFORMATION technology" OR DE "COMPUTER software" OR DE "COMPUTERS" OR DE "ELECTRONIC data processing" OR DE "COMPUTATIONAL linguistics"	Any field	16,141

24	19 OR 20 OR 21 OR 22 OR 23		84,205
25	18 AND 24		114
26	Limit 25 to English		109

Table 14. Linguistics and Language Behavior Abstracts via ProQuest, performed November 21, 2022.

Search	Query	Field	Results
1	troch* OR assonan* OR scansion* OR iamb* OR consonan* OR caden* OR prosod* OR rhythm* OR meter* OR metre* OR metric*	TITLE	9,190
2	troch* OR assonan* OR scansion* OR iamb* OR consonan* OR caden* OR prosod* OR rhythm* OR meter* OR metre* OR metric*	SUMMARY	29,399
3	troch* OR assonan* OR scansion* OR iamb* OR consonan* OR caden* OR prosod* OR rhythm* OR meter* OR metre* OR metric*	SUBJECT	15,523
4	1 OR 2 OR 3		33,506
5	music* OR song* OR poem* OR poet* OR lyric* OR verse* OR aria* OR hymn* OR choral* OR chorus* OR choir* OR tune*	TITLE	8,386
6	music* OR song* OR poem* OR poet* OR lyric* OR verse* OR aria* OR hymn* OR choral* OR chorus* OR choir* OR tune*	SUMMARY	20,975
7	music* OR song* OR poem* OR poet* OR lyric* OR verse* OR aria* OR hymn* OR choral* OR chorus* OR choir* OR tune*	SUBJECT	30,695
8	5 OR 6 OR 7		39,319
9	4 AND 8		3,332
10	rhyme* OR rhyming OR rime* OR riming OR quatrain* OR limerick* OR sonnet* OR couplet* OR stanza* OR ballad*	TITLE	885
11	rhyme* OR rhyming OR rime* OR riming OR quatrain* OR limerick* OR sonnet* OR couplet* OR stanza* OR ballad*	SUMMARY	3,835
12	rhyme* OR rhyming OR rime* OR riming OR quatrain* OR limerick* OR sonnet* OR couplet* OR stanza* OR ballad*	SUBJECT	1,364
13	MAINSUBJECT.EXACT.EXPLODE("Rhyme (Poetics)") OR MAINSUBJECT.EXACT.EXPLODE("Rhyme (Phonology)")	Anywhere	359
14	10 OR 11 OR 12 OR 13		4,423
15	9 OR 14		7,164
16	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	TITLE	10,410

17	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	SUMMARY	35,176
18	automat* OR algorithm* OR comput* OR unsupervised OR autoencod* OR "neural network*" OR software* OR "machine learning" OR "artificial intelligence" OR "probabilistic method*" OR "probabilistic model*" OR markov OR "graph theor*" OR "graph structure*" OR "constraint satisfact*" OR "constraint program*" OR "deep learning" OR "reinforcement learning"	SUBJECT	31,157
19	MAINSUBJECT.EXACT.EXPLODE("Computer Software") OR MAINSUBJECT.EXACT.EXPLODE("Computer Applications") OR MAINSUBJECT.EXACT.EXPLODE("Markov Models") OR MAINSUBJECT.EXACT.EXPLODE("Neural Networks") OR MAINSUBJECT.EXACT.EXPLODE("Constraints (Optimality Theory)") OR MAINSUBJECT.EXACT.EXPLODE("Machine Learning") OR MAINSUBJECT.EXACT.EXPLODE("Artificial Intelligence") OR MAINSUBJECT.EXACT.EXPLODE("Algorithms") OR MAINSUBJECT.EXACT.EXPLODE("Computational Linguistics")	Anywhere	4,017
20	16 OR 17 OR 18 OR 19		50,835
21	15 AND 20		412
22	Limit 21 to English		359

ACL Anthology, ACL Bibliography

Downloaded full anthology as BibTeX with abstracts on 21 November 2022

Filtered from the .bib file using a custom-built python script to find:

Rhyme* OR rhyming OR quatrain* OR couplet* OR ((music* OR song* OR poem* OR poet* OR lyric* OR verse* OR tune*) AND (Troch* OR assonance* OR scansion OR Iamb* OR Prosod* OR Limerick* OR Sonnet* OR consonance* OR cadence*))