AcrosticSleuth: Probabilistic Identification and Ranking of Acrostics in Multilingual Corpora

Aleksandr Fedchin,[1] Isabel Cooperman,[2] Pramit Chaudhuri,[3] Joseph P. Dexter[4,5,6]

[1]Department of Computer Science, Tufts University
[2]Department of Classical and Ancient Near Eastern Studies, University of Wisconsin-Madison
[3]Department of Classics, University of Texas at Austin
[4]Institute of Collaborative Innovation and Department of Computer and Information Science, University of Macau
[5]Roux Institute and Khoury College of Computer Sciences, Northeastern University
[6]Data Science Initiative and Department of Human Evolutionary Biology, Harvard University

aleksandr.fedchin@tufts.edu, icooperman@wisc.edu,
pramit.chaudhuri@austin.utexas.edu, jdexter@um.edu.mo

Abstract

For centuries, writers have hidden messages as acrostics, in which initial letters of consecutive lines or paragraphs form meaningful words or phrases. Scholars searching for acrostics manually can only focus on a few authors at a time and often favor qualitative arguments about whether a given acrostic is accidental or intentional. Here we describe AcrosticSleuth, a first-of-its-kind approach to identify acrostics automatically and rank them by the probability that the corresponding sequence of characters does not occur by chance. Since acrostics are rare, we formalize the problem as a binary classification task in the presence of extreme class imbalance. To evaluate AcrosticSleuth, we present the Acrostic Identification Dataset (AcrostID), a collection of acrostics from the WikiSource online database. Despite the class imbalance, AcrosticSleuth achieves F1 scores of 0.39, 0.59, and 0.66 on the French, English, and Russian subdomains of WikiSource, respectively. We further demonstrate that AcrosticSleuth can identify previously unknown instances of wordplay in high-profile literary contexts, including the English philosopher Thomas Hobbes' signature in the opening paragraphs of *The Elements of Law*.

1 Introduction

If you put together the initial letters of the 14 opening paragraphs of Thomas Hobbes' *The Elements of Law*, you will discover that they spell `THOMAS[OF]HOBBES`. Such hidden messages, in which initial letters of lines or paragraphs form a meaningful word or phrase, are called *acrostics*. Acrostics are easy to find if you know where to look—some authors even draw attention to them—but can otherwise be difficult to notice. For example, to the best of our knowledge we are the first to identify the Hobbes acrostic, despite its appearance at the beginning of an important, well-studied text by a famous author. The subtle, often playful nature of acrostics has kept the literary device in regular if infrequent use throughout the centuries. More recently, Russian dissidents have inserted anti-government messages as acrostics into mainstream publications.[1]

In contrast to these unambiguous examples, scholars have also argued for the intentionality of much shorter acrostics, such as the supposed acrostic `MARS` in the middle of Vergil's *Aeneid* (Fowler, 1983). Critics have seen the use of two regular Latin terms for war within the passage ("Martem," "bellum") as validating the acrostic. Such discussions, however, have not considered the probability of encountering the four-letter sequence.

In this paper, we introduce AcrosticSleuth, a tool that can identify putative acrostics in large corpora and rank them by the probability that the sequence of initial characters does not occur by chance (and therefore may have been inserted intentionally by the author). AcrosticSleuth is a command line tool available on GitHub under the MIT license.[2] From

[1]To cite two examples, politically persecuted film director and LGBT activist Kirill Serebrennikov (2020) encoded a message in his final speech to the court that spells НИОЧЕМНЕЖАЛЕЮСОЧУВСТВУЮВАМ ("I have no regrets. I am sorry for you."), and scholar Ilya Lemeshkin (Медиазона, 2023) published a paper in a government-funded journal with an acrostic СДОХНИПУТЛЕРНЕТВОЙНЕИЛ ("Die Putler. No to war. I.L.").

[2]https://github.com/acrostics/acrostic-sleuth

a statistical perspective, the acrostic identification problem presents a challenge in the form of extreme class imbalance: acrostics are very rare. In Section 3, we discuss how we identify and rank acrostics, as well as the implementation details that allow for efficient search.

To evaluate AcrosticSleuth, we create the Acrostic Identification Dataset (AcrostID), a collection of labeled acrostics from the English, French, and Russian subdomains of the WikiSource database of texts. The dataset is available under the MIT license and includes all acrostics that have been explicitly referred to or formatted as such on WikiSource.[3] We show that AcrosticSleuth successfully identifies acrostic poems and achieves F1 scores of 0.39, 0.59, and 0.66 on French, English, and Russian corpora, respectively. In Section 4, we present these results, provide a comparison of the tool's performance across languages, and discuss acrostics found by AcrosticSleuth that have not been recognized previously.

2 Background

Some prior work has applied quantitative methods to investigate the intentionality of acrostics in Shakespeare (Eckler, 1985) and Horace (Morgan, 1993), among other authors. Such studies typically consider the probabilities of individual acrostics in isolation (Morgan, 1993), instead of ranking them in comparison to other candidates. The basic limitation of this approach, as noted by Robinson (2019), is that, while the probability of any given acrostic is indeed very low, one is nevertheless almost guaranteed to stumble upon some accidental acrostic in a sufficiently long text. The lottery offers a good analogy: the chances of winning are extremely low for any one person, but someone still takes home the jackpot.

Our work falls under the broader category of automated analysis of wordplay and puzzles, with AcrosticSleuth similar in several respects to crossword solving tools (Kulshreshtha et al., 2022). On the generative side, there has been substantial work on language models that can compose acrostic poems (Agarwal and Kann, 2020; Shen et al., 2019), paraphrase existing texts to introduce acrostics (Stein et al., 2014), produce anagrams (Jordan and Monteiro, 2003), or synthesize other types of wordplay (Liu et al., 2020). The wide availability of such tools, and the drive for creative language encodings intended to avoid censorship in online communication (Ji and Knight, 2018), suggest that acrostics may become even more widespread in the future.

3 Methods

In this section, we outline our approach for enumerating and ranking candidate acrostics. We define the problem as follows: given a sequence of line-initial characters in a text, rank all possible subsequences based on the probability that they come from natural speech and have not been selected at random from the distribution of initial letters. Our hypothesis is that this probability should reflect intentionality: the higher the probability, the more likely it is that the corresponding characters have been deliberately made to form meaningful words or phrases by the author.

Consider the binary classification problem of labeling a sequence of characters as "acrostic" or "not an acrostic." This problem suffers from extreme class imbalance: most sequences of characters will not be acrostics. Since we do not know the a priori probability $P(a)$ of encountering an acrostic, we cannot directly compute $P(a|s)$, the probability that a given sequence of characters s is an acrostic. To rank candidate acrostics, however, it is sufficient to estimate $\frac{P(a|s_1)}{P(a|s_2)}$, the ratio of two such probabilities for two different strings s_1 and s_2. By Bayes' theorem, this ratio is equal to $\frac{P(s_1|a)P(a)P(s_2)}{P(s_1)P(s_2|a)P(a)} = \frac{P(s_1|a)P(s_2)}{P(s_1)P(s_2|a)} = \frac{P(s_1|a)(P(s_2|a)P(a)+P(s_2|\neg a)P(\neg a))}{P(s_2|a)(P(s_1|a)P(a)+P(s_1|\neg a)P(\neg a))}$. We now posit that $P(a)$, the a priori probability of encountering an acrostic, must be very small. As $P(a)$ approaches zero, the ratio we need to estimate approaches $\frac{P(s_1|a)P(s_2|\neg a)}{P(s_1|\neg a)P(s_2|a)}$. In other words, computing $\frac{P(s|a)}{P(s|\neg a)}$ for every string s gives us a metric by which we can rank all candidate acrostics.

Of the two probabilities involved in computing the rank, estimating $P(s|\neg a)$ for some s is trivial—it is the conditional probability of first letters in each line forming the sequence of characters s under the assumption that the poem contains no acrostics. When there are no acrostics, each character in s is drawn independently at random from the overall distribution of line-initial characters in the target language, so that $P(s|\neg a)$ is a product of such probabilities for individual characters. On the other hand, $P(s|a)$, the probability of encountering

[3] https://github.com/acrostics/acrostic-identification-dataset

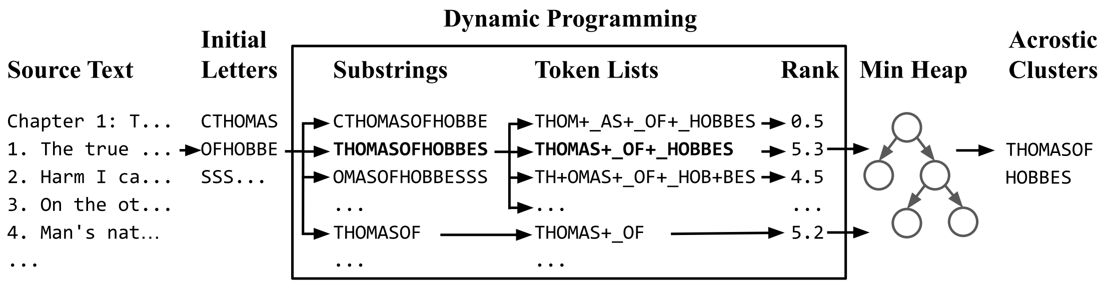


Figure 1: Workflow schematic for AcrosticSleuth.

a sequence of characters in an acrostic, is more difficult to estimate.

In this study, we assume that acrostics are similar to regular text, which allows us to use pre-trained language models. AcrosticSleuth relies on unigram language models produced by SentencePiece (Kudo and Richardson, 2018) to estimate the probability that an acrostic spells some given sequence of characters. SentencePiece is an unsupervised text tokenizer, in which subword-level tokens are chosen to fit the vocabulary size specified by the user. SentencePiece has several advantages for the present application: it is fully unsupervised and can be adapted to multiple languages; it uses subword tokens and therefore can handle out-of-vocabulary words, such as names, which often feature in acrostics; and it requires minimal compute. We found that AcrosticSleuth achieves best performance when using language models with large numbers of tokens (see Appendix).

Figure 1 illustrates the workflow of AcrosticSleuth, using Hobbes' *The Elements of Law* as an example. AcrosticSleuth first converts the source text into a string of initial letters. This step involves minimal language-specific preprocessing, such as removing all non-alphabetic characters. Next, AcrosticSleuth considers every possible substring of initial letters up to some fixed length and, for each substring, every possible way to tokenize it. For each resulting list of tokens, AcrosticSleuth computes the rank using dynamic programming; we process substrings that end earlier in the text first and store the highest ranking tokenization of previously encountered substrings. Note that the use of a unigram model simplifies the dynamic programming setup substantially, since the probability assigned to the next token is independent from the previous ones. We further boost performance by supporting multithreading and maintaining a cache of commonly occurring substrings.

AcrosticSleuth uses a min-heap data structure to keep track of the highest ranking candidate acrostics it has encountered so far. The size of the heap is fixed and can be specified by the user. When reporting the results, AcrosticSleuth aggregates candidate acrostics that overlap into one result and uses the top candidate to rank the whole cluster. For example, in Figure 1 both THOMASOF and THOMASOFHOBBES end up in the min-heap as high-ranking candidate acrostics, but AcrosticSleuth reports them as a single cluster because they overlap.

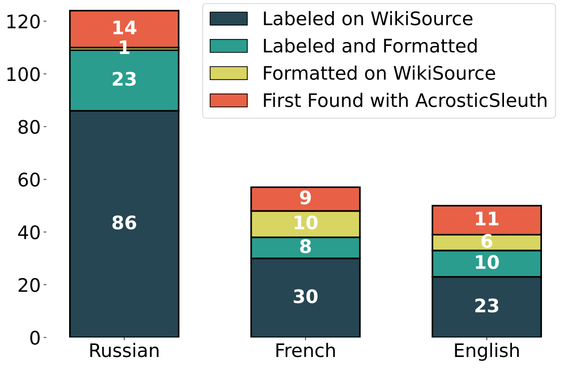


Figure 2: Categorization of the methods by which the acrostics in AcrostID were identified. Acrostics first found using AcrosticSleuth are excluded when computing recall discussed below.

4 Results and Evaluation

4.1 Acrostic Identification Dataset

To evaluate AcrosticSleuth's performance, we create the Acrostic Identification Dataset (AcrostID), which is comprised of acrostics found on WikiSource, a Wikipedia-affiliated online library of literature, parliamentary proceedings, and other texts. We chose Wikisource for several reasons, including its multilingual and cross-genre coverage, as well as the availability of partial annotations (many texts are explicitly labeled or formatted as acrostics). To obtain a set of "true" acrostics on which we can evaluate our tool, we perform the following two tasks. First, we manually inspect all

Acrostic	WikiSource Page
TOJOSEPHKNIGHT	Page:Notes by the Way.djvu/61
IESUCHRISTSONNEOFGODTHESAVIOR	Page:Whole prophecies of Scotland, England, Ireland, France & Denmark.pdf/46
PRINCECHARLIE	Page:Carroll - Three Sunsets.djvu/83
CORNELIABASSET	Ben King's Verse/Asphodel
KATHLEENBRUCE	Page:Clouds without Water (Crowley, 1909).djvu/24
AMAZING	Page:Amazing Stories Volume 17 Number 06.djvu/6
PERHAPS	Page:Love's trilogy.djvu/79
ALICEPLEASANCELIDDELL	Page:Complete Works of Lewis Carroll.djvu/292
THOMAS[OF]HOBBES	The Elements of Law/Part I/Chapter 1
MARYSTOKES	Page:Notes and Queries - Series 12 - Volume 4.djvu/257
SURVIVAL	United States Army Field Manual 7-93 Long-Range Surveillance Unit Operations

Table 1: English acrostics that were not labeled or formatted as such on WikiSource as of April 20, 2024.

uses of the word "acrostic" on WikiSource ("акростих" in Russian, "acrostiche" in French). In cases in which the word refers to specific lines, we mark these as acrostics. Using this method, we identify 33 acrostics in English WikiSource, 109 in Russian, and 38 in French.

We also include acrostics based on formatting: initial letters of an acrostic are often highlighted in bold or in red or are rotated by 90 degrees. We look at all sequences of five or more consecutive lines in which initial letters are specially formatted and identify cases in which the initial letters form one or more words in the source language. Finally, we manually inspect the corresponding WikiSource pages to confirm that the formatting is not accidental. This method allows us to identify a further six English acrostics, as well as one Russian and 10 French. Figure 2 summarizes the sources of acrostics in AcrostID, as well as the number of new acrostics identified by AcrosticSleuth, for each language. When reporting the number of acrostics above, we count some acrostics together as one entry (e.g., when a single acrostic is split between two WikiSource pages or reproduced multiple times), so as to enable fairer evaluation of tool performance.

4.2 Experiments

We perform all experiments on an M3 MacBook Pro with 48 GB RAM and 16 CPU cores. All experiments can be reproduced in under an hour using comparable resources. Figure 3 summarizes the performance of AcrosticSleuth on AcrostID. In each subplot, the y-axis shows the tool's performance, and the x-axis (logarithmic scale) indicates the number of first-ranking results for which the corresponding metric is calculated. Despite the extreme class imbalance, AcrosticSleuth achieves top F1 scores of 0.39, 0.59, and 0.66 on French, English, and Russian corpora, respectively, suggesting that AcrosticSleuth can be used for quick identification of most acrostics in a given corpus.

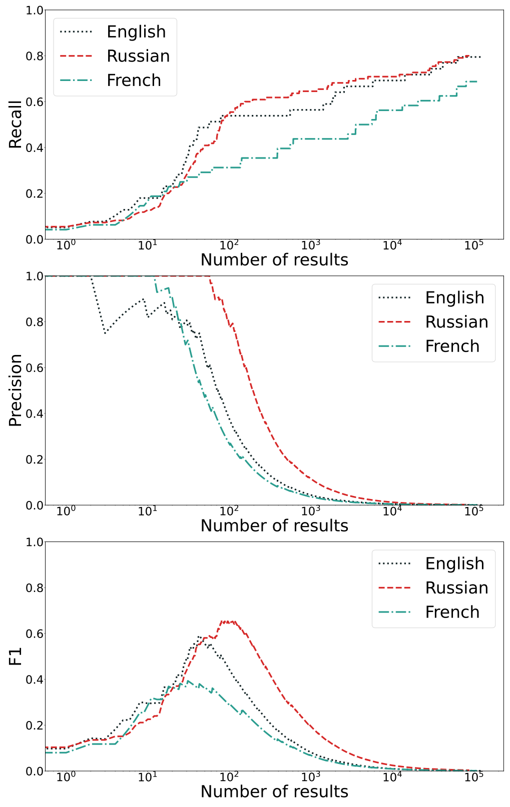


Figure 3: Recall, precision, and F1 score of AcrosticSleuth.

Figure 3 also shows that AcrosticSleuth achieves the best performance on the Russian corpus and the worst on the French; this difference may be due to the composition of the datasets themselves. In particular, Russian WikiSource contains a large number of acrostics from 17th and 18th century texts, which tend to span dozens of lines and are thus easier to identify, whereas many acrostics in French WikiSource are split across multiple pages or are otherwise more difficult to find due to formatting issues.

When calculating recall, we only consider as acrostics those instances that were identified manually, as described above. When calculating precision, however, we also take into account acrostics that the tool identifies but that are not labeled or formatted as such on WikiSource. To this end, we manually inspect the top 1000 results AcrosticSleuth returns for each language and note those we believe to be acrostics beyond any reasonable doubt (Figure 2). Table 1 lists all unlabeled and unformatted acrostics found in the English subdomain of WikiSource. Some acrostics in Table 1 have been identified before, such as the one by Lewis Carroll, although the corresponding page on WikiSource contains no reference to the acrostic (as of April 20, 2024). Other acrostics, however, are new discoveries, such as the `THOMAS[OF]HOBBES` example discussed in the Introduction.

To explore the capabilities of AcrosticSleuth for multilingual and diachronic analysis, we also run it on Musisque Deoque and Poeti D'Italia, two databases of Latin poetry. Among the results is the acrostic `ARSPOETICA` in a 14th century poem by Albertino Mussato, which was only noticed for the first time in 2022 (Hosle, 2022). The poem laments that the Italy of Mussato's time is not safe for poets, and the acrostic identifies the female subject of the opening sentence, suggesting that she (the art of poetry) is not at ease.

5 Discussion and Future Work

Throughout this paper, we discuss the acrostics that our tool identifies only insofar as they are relevant for the tool's evaluation. A direction for future qualitative research is the analysis of acrostics in their own right. A preliminary scan of our dataset reveals two tendencies that are worth discussing. First, the majority of acrostics we find encode names, usually as a form of dedication. Second, the majority of acrostics we find appear at the very beginning of their respective texts. These observations have implications for the future development of AcrosticSleuth and the study of acrostics in general.

The abundance of names in acrostics complicates their detection because many names are transliterations from another language, and a language model is naturally biased towards most commonly occurring names and syllables. For instance, compare the log probability of -20 that our English model assigns to "Walter Scott" to the -44 log probability that it assigns to "Amos Tutuola," despite both names being the same length. This difference is not due simply to the Yoruba name "Tutuola" being an out-of-vocabulary word—it persists even if we use a smaller 900-token model, which does not have any of the four names in the vocabulary but instead splits them into subtokens (`a-mo-s t-ut-u-o-la` and `w-al-ter sc-ot-t`). In the future, we may experiment with biasing AcrosticSleuth toward the vocabulary of the specific text under analysis.

The tendency of acrostics to appear in the opening lines of a text is not surprising—this is where readers are most likely to look for them—but it does highlight the idiosyncrasy of examples such as the `MARS` acrostic we mention in the Introduction, which appears in the middle of the *Aeneid*. We have run AcrosticSleuth on Musisque Deoque and Poeti D'Italia, two databases of Latin poetry, and find that the tool ranks this acrostic very low—you can locate comparable examples in virtually any text that is a few hundred lines long. Coupled with its position in the middle of the poem, rather than at the very beginning, this finding gives little reason to consider the acrostic intentional from a statistical perspective. Our point here, however, is not necessarily to argue against the intentionality of this specific instance, but rather to emphasize the complementary nature of qualitative and quantitative analysis. The results returned by AcrosticSleuth should be carefully examined by a specialist, and, conversely, every conjecture should be reviewed from a statistical point of view.

6 Conclusion

This paper presents AcrosticSleuth, a tool for identifying and ranking acrostics in large corpora of texts. Using a new benchmark dataset (AcrostID), we show that the tool can identify known acrostics in French, English, and Russian corpora with high recall. In addition, we use AcrosticSleuth to uncover important acrostics that have not been discussed previously by literary scholars, such as Hobbes' `THOMAS[OF]HOBBES`.

Acknowledgements

We thank Kinch Hoekstra and Johann Sommerville for their insights regarding Hobbes' acrostic, Jeff Foster for his feedback at an early stage of this project, and the anonymous reviewers for their helpful suggestions.

Limitations

The present implementation and evaluation of AcrosticSleuth has several limitations, which represent promising avenues for future work. First, we use only a single language model, a unigram model from SentencePiece. Although well-justified and effective for our application, future work on acrostic detection should characterize potential performance gains from using a large language model in place of SentencePiece. Second, while we perform a multilingual evaluation of AcrosticSleuth involving three languages (French, English, and Russian), it would be valuable to extend the coverage of AcrostID more broadly, especially in light of the language-specific performance differences we observe. Finally, as suggested by our case study with Mussato, Latin literature presents an interesting object for further study. Roman authors composed not only regular acrostics but also telestics (formed by combining the final letters of each line), mesostics (formed by every n-th letter), and diagonal acrostics. In addition to completing a systematic study of conventional Latin acrostics, in future work we also plan to extend the capabilities of AcrosticSleuth to handle such alternative forms of wordplay.

Ethical Considerations

In the Introduction, we discuss how acrostics have been used by dissidents to insert anti-war messages into mainstream media. In theory, one can imagine a malicious actor using a tool such as AcrosticSleuth to screen incoming publications for "undesirable" acrostics. In practice, however, AcrosticSleuth can only identify one specific kind of acrostic—those formed by initial letters of each line or paragraph. Other kinds of acrostics, such as those formed by initial letters of each word or sentence, would remain undetected. Even if the tool's functionality were further extended to cover these cases as well, one could always come up with a new way to encode a hidden message into the text. The point of acrostics by Lemeshkin and Serebrennikov, for instance, is precisely that there is no way to silence or prevent such artistic expressions of opinion, regardless of how much effort one spends on censorship. These types of examples, moreover, are not hidden in the sense of wishing to evade all notice—they are calculated to provoke or amuse, with Lemeshkin publicly revealing his acrostic right after publication. Hence, in the unlikely event that AcrosticSleuth were to be used for censorship purposes, we believe that the effort would prove futile.

References

Rajat Agarwal and Katharina Kann. 2020. Acrostic poem generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1230–1240, Online. Association for Computational Linguistics.

A. Ross Eckler. 1985. Are Acrostic Messages Real? *Word Ways*, 18(3):187–189.

Медиазона. 2023. С сайта журнала МГИМО пропала публикация ученого из России, который зашифровал в научной статье слова «Сдохни, Путлер. Нет войне». Медиазона. Last accessed: 2024-08-05.

Don P. Fowler. 1983. An Acrostic in Vergil (*Aeneid* 7. 601–4)? *Classical Quarterly*, 33(1):298–298.

Paul K. Hosle. 2022. An 'Ars Poetica' acrostic in a poem of Albertino Mussato. *Arctos*, 56:27–31.

Heng Ji and Kevin Knight. 2018. Creative language encoding under censorship. In *Proceedings of the First Workshop on Natural Language Processing for Internet Freedom*, pages 23–33, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Timothy R. Jordan and Axel Monteiro. 2003. Generating anagrams from multiple core strings employing user-defined vocabularies and orthographic parameters. *Behavior Research Methods, Instruments, & Computers*, 35(1):129–135.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Saurabh Kulshreshtha, Olga Kovaleva, Namrata Shivagunde, and Anna Rumshisky. 2022. Down and across: Introducing crossword-solving as a new NLP benchmark. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2648–2659, Dublin, Ireland. Association for Computational Linguistics.

Yusen Liu, Dayiheng Liu, and Jiancheng Lv. 2020. Deep poetry: A Chinese classical poetry generation system. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(09):13626–13627.

Gareth Morgan. 1993. *Nullam, vare ...* Chance or Choice in *Odes* 1.18? *Philologus*, 137(1):142–145.

Matthew Robinson. 2019. Arms and a mouse: approaching acrostics in Ovid and Vergil. *Materiali e discussioni per l'analisi dei testi classici*, 82(1):23–73.

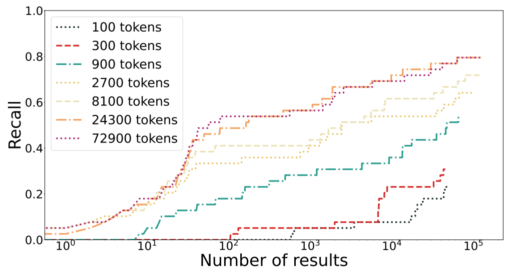
Kirill Serebrennikov. 2020. «Ни о чем не жалею. Сочувствую вам». Новая газета. Last accessed: 2024-08-05.

Liang-Hsin Shen, Pei-Lun Tai, Chao-Chung Wu, and Shou-De Lin. 2019. Controlling sequence-to-sequence models - a demonstration on neural-based acrostic generator. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 43–48, Hong Kong, China. Association for Computational Linguistics.

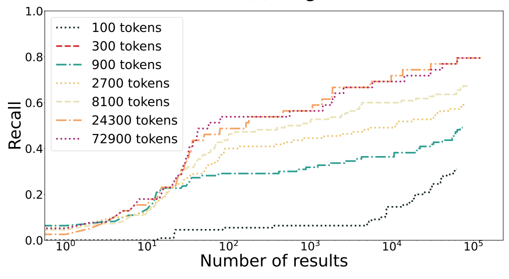
Benno Stein, Matthias Hagen, and Christof Bräutigam. 2014. Generating acrostics via paraphrasing and heuristic search. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2018–2029, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Appendix

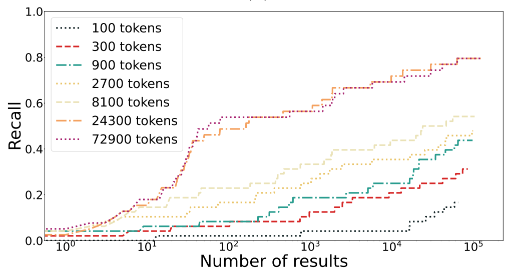
Effect of Model Size on Performance



(a) English



(b) Russian



(c) French

Figure 4: Effect of language model size on AcrosticSleuth's recall for English, Russian, and French corpora.

In Section 3, we write that AcrosticSleuth reaches peak performance when using SentencePiece language models with the largest number of tokens. Figure 4 illustrates this point further by showing the recall that AcrosticSleuth achieves for different languages with models of different sizes. Note that the performance of the largest two models (72900 and 24300 tokens, respectively) is very similar, suggesting that further increases to the model's size would yield diminishing returns.