SOFTMATCHA: A SOFT AND FAST PATTERN MATCHER FOR BILLION-SCALE CORPUS SEARCHES

Hiroyuki Deguchi[1,*], **Go Kamoda**[2], **Yusuke Matsushita**[3], **Chihiro Taguchi**[4], **Kohei Suenaga**[3], **Masaki Waga**[3], **Sho Yokoi**[5,2,6]
[1]NAIST, [2]Tohoku University, [3]Kyoto University, [4]University of Notre Dame,
[5]NINJAL, [6]RIKEN
deguchi.hiroyuki.db0@is.naist.jp, go.kamoda@dc.tohoku.ac.jp,
ymat@fos.kuis.kyoto-u.ac.jp, ctaguchi@nd.edu
ksuenaga@kuis.kyoto-u.ac.jp, mwaga@fos.kuis.kyoto-u.ac.jp
yokoi@ninjal.ac.jp

ABSTRACT

Researchers and practitioners in natural language processing and computational linguistics frequently observe and analyze the real language usage in large-scale corpora. For that purpose, they often employ off-the-shelf pattern-matching tools, such as `grep`, and keyword-in-context concordancers, which is widely used in corpus linguistics for gathering examples. Nonetheless, these existing techniques rely on surface-level string matching, and thus they suffer from the major limitation of not being able to handle orthographic variations and paraphrasing—notable and common phenomena in any natural language. In addition, existing continuous approaches such as dense vector search tend to be overly coarse, often retrieving texts that are unrelated but share similar topics. Given these challenges, we propose a novel algorithm that achieves *soft* (or semantic) yet efficient pattern matching by relaxing a surface-level matching with word embeddings. Our algorithm is highly scalable with respect to the size of the corpus text utilizing inverted indexes. We have prepared an efficient implementation, and we provide an accessible web tool. Our experiments demonstrate that the proposed method (i) can execute searches on billion-scale corpora in less than a second, which is comparable in speed to surface-level string matching and dense vector search; (ii) can extract harmful instances that semantically match queries from a large set of English and Japanese Wikipedia articles; and (iii) can be effectively applied to corpus-linguistic analyses of Latin, a language with highly diverse inflections.

1 INTRODUCTION

Recent advances in natural language processing (NLP) and corpus linguistics are largely driven by the availability of massive text corpora (Gao et al., 2020; Biderman et al., 2023b; Raffel et al., 2019; Van Der Zwaan et al., 2017; Mcgillivray et al., 2020). The field of NLP, driven by the grand goal of building intelligent chatbots and machine translation systems, has achieved remarkable breakthroughs over the past decade, largely thanks to self-supervised representation learning from vast corpora (Mikolov et al., 2013; Pennington et al., 2014; Devlin et al., 2019a; Radford et al., 2019). Notable causal language models (LMs), capable of passing high-stakes exams (Katz et al., 2024; OpenAI, 2024; Jung et al., 2023), serving as general-purpose problem solvers (Brown et al., 2020), and engaging in realistic conversations with beloved characters (De Freitas, 2023; Chen et al., 2024), owe their foundational abilities to next-token prediction learned from large-scale data (Brown et al., 2020; Dubey et al., 2024; Riviere et al., 2024). In corpus linguistics (McEnery & Hardie, 2011), computational linguistics (Jurafsky & Martin, 2024), and digital humanities (Jensen, 2014)—disciplines aiming to uncover the scientific and computational principles of human language—such vast linguistic resources, consisting of the language use itself, have become more indispensable than ever in this era of large-scale corpora (Van Der Zwaan et al., 2017; Mcgillivray et al., 2020).

[*]Currently affiliated with NTT. E-mail: hiroyuki.deguchi@ntt.com

Table 1: *Hard* pattern matching (e.g., `grep`) and *soft* pattern matching (our method). Hard matching is based on *surface-level* comparison and does not match if, e.g., a synonym is used; Soft matching is based on *semantic* comparison and robust to such variations thanks to word embeddings.

Pattern	**Theorem 1**	**John was born in**
Hard matching (e.g., `grep`)	··· thanks to **Theorem 1** ···	··· **John was born in** 1345 ···
Soft matching (**ours**)	··· thanks to **Theorem 1** ···	··· **John was born in** 1345 ···
	··· **Theorem 3** holds because ···	··· **Edward had died in** May 1910 ···
	··· By **Lemma 5**, we may assume ···	··· **Robert was born in** England ···
	··· **Equation 1** describes ···	··· the Emperor **Henry, died in** 1125 ···

Given this context, the demand for efficient pattern matchers that enable rapid searches across massive corpora is higher than ever (Liu et al., 2024; Smadja, 1993). For example, when harmful information, misinformation, or memorized privacy information is generated by a large LM, it is necessary to identify the corresponding training instances where this information originated (Guo et al., 2022; Wang et al., 2024c; Ippolito et al., 2023; Biderman et al., 2023a). Another example is when researchers wish to determine which of two linguistic phenomena of interest is more frequent; conducting exhaustive searches across the largest available corpora is essential (Biber, 2015) for this purpose. Notably, many linguistic phenomena—word frequency as a simple example—follow a power-law distribution (Zipf, 1951; Heap, 1980; Kobayashi & Tanaka-Ishii, 2018). Consequently, only with vast corpora can we observe and analyze the various rare events residing in the long tail, which constitute the majority of linguistic phenomena.

One of the challenges when working with large corpora is that pattern-matching tools based on string matching (Hakak et al., 2019), such as `grep` (Bambenek & Klus, 2009) and `ripgrep` (Gallant, 2024), or linguist-oriented tools often referred to as KWIC (Anthony, 2013; Culy & Lyding, 2010; Schweinberger, 2024), primarily rely on surface-level exact matching as their core strategy. Because natural languages are characterized by their flexibility and richness in how humans can express similar concepts in different ways (McKeown, 1979; Witteveen & Andrews, 2019; Ganitkevitch et al., 2013), strictly exact string matching may not meet users' demands. For example, on top of the query word in standard spelling, it is often desirable to catch non-standard spellings as well, such as *how r u* instead of *how are you*, which are widespread particularly on the web and in texting (Schulz, 2018). Additionally, it is also desirable to catch different inflected word forms such as *sing*, *sang*, *sung*, *sings*, and *singing*, which differ only in their morphological features and share the same lemma (base form) (Don, 2014; Embick, 2015). Rule-based exact matching is particularly hard when the target language is morphologically complex and exhibits irregular inflectional patterns.

To resolve the mismatch between the symbolic nature of existing pattern matchers and the diverse orthographic and morphological variations inherent in natural language, we have developed a *soft* (or semantic) yet efficient pattern matcher (Section 3). The core strategy is based on the simple idea of *softening* the matching process from binary $\{0, 1\}$ values to continuous values, using word embeddings. By adopting inverted indexes and several other techniques, our tool can enumerate all softly matching instances in billion-scale corpora in less than a second. Table 1 shows specific examples of its operation. Our proposed method is capable not only of enumerating exact matches but also of flexibly listing semantically similar instances, even when their surface forms differ. For example, given the query "*Theorem 1*", the method can retrieve instances such as "*Lemma 5*". This characteristic substantially enhances key tasks in both NLP and corpus linguistics. For instance, it improves the filtering of harmful texts, and facilitates more efficient example retrieval, particularly for languages with complex morphological features (Section 4).

The contributions of our study are summarized as follows.

- We developed a *soft* (or semantic) pattern-matching algorithm—a relaxation of the exact string matching—using word embeddings (Section 3). By leveraging inverted indexing, we achieved high scalability (Sections 2.1, 2.3 and 3.3).[1]

[1] GitHub: https://github.com/softmatcha/softmatcha

- We developed an easy-to-use web demo to facilitate interaction with the soft matching algorithm.[2] This demo is particularly beneficial for NLP researchers and developers who want to analyze LMs in a data-driven manner, as well as for language learners and humanities researchers who are conducting language analysis on large corpora from the perspectives of corpus linguistics and digital humanities (Section 4.1)
- For quantitative evaluation, we verified that our method achieves complete enumeration in less than a second on billion-scale corpora, which are commonly used in training large LMs (Section 4.2). For instance, the running time for searching over an English corpus with 3.4B words was less than 0.1 seconds without GPUs. This performance is comparable in speed to dense vector search; moreover, our method enables the retrieval of examples closely aligned with specific queries, rather than just broad topical similarities.
- We conducted qualitative evaluations in specific scenarios to verify the usefulness of the proposed method in both the fields of NLP and corpus linguistics. In experiments assuming the training of LMs on large-scale corpora, we provided search examples for identifying and removing harmful instances contained within the corpus (Section 4.2). In experiments assuming the analysis of classical Western languages by linguists, we chose Latin—a language with highly complex inflections—as an example. We demonstrated that a corpus search with our tool can flexibly extract morphologically and semantically similar usage examples from the corpus (Section 4.3).

2 Related work

The procedures for finding sentences (or lines, documents) that match a given pattern (query) are prevalent across nearly all areas of computer science and data science, making it difficult to enumerate all related research. In this section, we review particularly relevant work in NLP and computational linguistics, which are the primary focus of this study, as well as in the closely related areas of string matching. Beyond the fields discussed here, we believe many other domains, such as information retrieval, time series analysis, and semantic web, also have connections to this research.

2.1 Natural language processing and corpus search

The significant progress in NLP over the past decade is largely attributed to deep-learning–based self-supervised representation learning (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017; Devlin et al., 2019a; Radford et al., 2019; Brown et al., 2020; Dubey et al., 2024). **The vast raw corpora** without label annotations serve as the source of such models' capabilities (Gao et al., 2020; Biderman et al., 2023b; Raffel et al., 2019), and, therefore, these corpora are continually referenced and analyzed for model improvement and evaluation (Biderman et al., 2023b). For instance, as LMs can memorize and elicit facts written in training corpora, they are reported to generate text containing privacy-related information (Huang et al., 2022; Li et al., 2023; Lukas et al., 2023) or content that could be used for terrorism or violence (Gehman et al., 2020; Schick et al., 2021; Kumar et al., 2023). In this context, Ippolito et al. (2023) work on filtering out verbatim memorization, requiring searches accross large-scale training corpora.[3] **N-gram substring pattern matchers**, as representative tools for corpus search in NLP, have been particularly well-developed even before the advent of deep learning (Sekine, 2008; Sekine & Dalwani, 2010). In terms of data structures, inverted indexes (Sekine & Dalwani, 2010; Rogozinski & Kuc, 2016), and suffix arrays including their variants (Sekine & Dalwani, 2010; Yamamoto & Church, 2001; Liu et al., 2024; Burrows et al., 1994; Ferragina & Manzini, 2005; Langmead et al., 2009) are typically employed. Our algorithm is based on inverted indexes due to its algorithmic requirements, which we discuss in Section 3. Here, regardless of which data structure is used, it is important to note that existing n-gram matching techniques assume exact surface-level matching, making it extremely challenging to search and enumerate all relevant sentences while handling the complex characteristics of natural language, such as paraphrasing, orthographic variation, and inflection. **Dense vector search** (Khattab & Zaharia, 2020; Karpukhin et al., 2020; Izacard et al., 2022; Wang et al., 2024a) has recently gained widespread popularity as the foundational technology for retrieval-augmented generation

[2] Demo: https://softmatcha.github.io. The screenshot of the demo is presented in Appendix C.

[3] Measuring influence of training corpus is also an important theme (Koh & Liang, 2017; Pruthi et al., 2020; Chen et al., 2021; Isonuma & Titov, 2024), but their application to vast corpora remains highly challenging.

(RAG) (Lewis et al., 2020; Khandelwal et al., 2020; Guu et al., 2020; Izacard et al., 2024). Dense vector search is highly compatible with approximate nearest neighbor search (Malkov & Yashunin, 2020; Jégou et al., 2011), and it offers a significant advantage in reducing the issue of hallucination (Ayala & Bechard, 2024; Gao et al., 2023) in scenarios requiring factual knowledge. However, dense vector search is a relatively "coarse" method, primarily used to retrieve documents that are topically similar, making it less suitable for the cases that require more specific n-gram level queries.

2.2 Corpus linguistics and corpus search

Corpus linguistics is a subfield of linguistics that utilizes corpora to study language based on examples of 'real-life' language use (McEnery & Wilson, 2001). Digital humanities, a closely related field, is an interdisciplinary domain that aims to advance humanities through the application of data and information science. In digital humanities as well, quantitative analysis of texts using corpora is also actively pursued (Jensen, 2014). **Corpora** consist of collections of texts or spoken language data (Van Der Zwaan et al., 2017; Mcgillivray et al., 2020), offering real-world examples of how language is used in various contexts. In most settings, an ideal corpus is large in size to ensure the statistical reliability of certain linguistic phenomena and to cover a broader range of language use, including rare occurrences such as low-frequency words (Ha et al., 2009; Coole et al., 2020). **Search tools** (Hockey & Martin, 1987; TEI Consortium, 2023) are an indispensable element of corpus linguistics because they enable the identification of linguistic patterns, such as word frequencies, collocations, and syntactic structures. Traditional search methods commonly used in corpus linguistics include exact matching and its extension with regular expressions (Jurafsky & Martin, 2024). However, given the nature of natural languages—with their morphological complexity and unlimited paraphrases and synonyms with varying surface forms—rule-based search methods face extreme difficulties in exhaustively enumerating instances that closely match a given query.

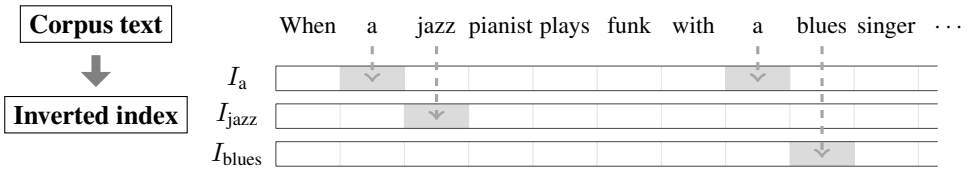
2.3 String matching

From an algorithmic point of view, our method is closely related to *string matching* algorithms (Hakak et al., 2019). In what follows, we briefly give an algorithmic comparison with some of them and elucidate their relationship to ours. **Offline string matching** algorithms process the corpus text to build an index that accelerates future searches, a technique widely applied in search engines and information retrieval systems such as Elasticsearch (Rogozinski & Kuc, 2016) and Apache Lucene (Grand et al., 2020). Inverted indexing is a well-known approach in this domain (Zobel & Moffat, 2006), and our method serves as a *soft* generalization of a string matching algorithm based on inverted indexing. **Online string matching** algorithms do not rely on preprocessing and handle the corpus text on the fly. Efficient algorithms include the Knuth-Morris-Pratt, Karp-Rabin, and Boyer-Moore algorithms (Knuth et al., 1977; Karp & Rabin, 1987; Boyer & Moore, 1977). While widely used in tools such as `grep`, online string matching is also a basis of runtime verification (Bartocci et al., 2018), where system's execution data is monitored. Although it is possible to relax these algorithms with word embeddings, the number of *soft* word comparisons in such a relaxation would be linear in the size of the corpus. In contrast, our algorithm requires only constant *soft* word comparisons, thanks to indexing. See Section 3.3 for the complexity analysis. **Approximate string matching** allows flexible matching, typically using edit distance to accommodate noisy or incomplete data (Navarro, 2001). Tools like `agrep` (Wu & Manber, 1992b;a) perform online matching that tolerates mismatches. Indexing-based algorithms are also proposed for approximate string matching (Boytsov, 2011). As a relaxation of exact string matching, approximate string matching is orthogonal to ours: approximate string matching focuses on relaxing surface-level comparison, while our approach focuses on semantic-level similarity based on word embeddings. For example, morphologically irregular forms such as "person" and "people", only differing in plurality, will likely be missed in approximate string matching, while our methods are able to catch them.

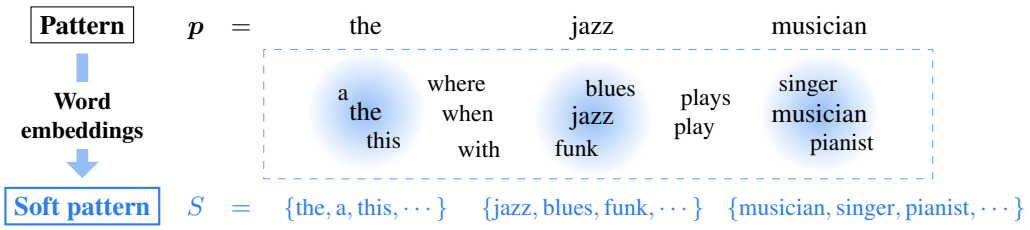
3 Our algorithm for soft pattern matching

3.1 Overview

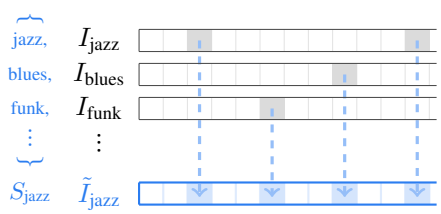
Key aspect of design: Hard vs. Soft. One key aspect we considered in designing the algorithm for soft pattern matching is *hard computation vs. soft computation*. *Hard* (or exact) pattern matching



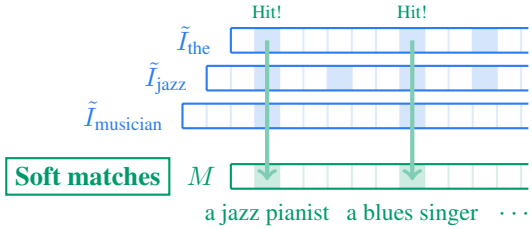
(a) Preprocessing: Build the inverted index of the corpus text.



(b) Matching Step 1: Soften the pattern into the soft pattern.



(c) Matching Step 2-1: Get the soft inverted index.



(d) Matching Step 2-2: Find the soft matches.

Figure 1: Illustration of our algorithm for soft pattern matching.

(such as `grep`) can process large texts blazingly fast. This is because hard matching requires only "*hard* computation", e.g., bitwise comparison, which is highly efficient. *Soft* pattern matching, on the other hand, involves "*soft* computation", e.g., cosine similarity of two vectors. Since such *soft* computation involves many floating-point arithmetic operations over high-dimensional vectors, it is typically much slower than *hard* computation. For instance, just taking the cosine similarity of a pair of word embeddings can be as expensive as thousands of simple bitwise operations.

Overview of our algorithm. With this in mind, we designed the algorithm for soft pattern matching, which is precise and efficient. Our algorithm is illustrated in Figure 1. Our key idea is to run soft computation *only over the vocabulary*, not over the whole text as a naive algorithm would do. More specifically, our algorithm works in the following two steps: For each query, (Step 1) it first performs *soft* computation over the *vocabulary*, *softening* the pattern into the *soft pattern* $S_1 S_2 \cdots S_n$ (Fig. 1b); (Step 2) then it performs *hard* computation that finds the *exact* positions in the corpus text that match the soft pattern (Figs. 1c and 1d). Here, the soft pattern $S_1 S_2 \cdots S_n$ is the key data in play. It consists of the set of vocabulary words that *softly* match each word of the pattern in terms of word embeddings. The amount of soft computation (Step 1) is small, given that the vocabulary size is typically much smaller than the size of the corpus text. The amount of hard computation (Step 2) is also quite small, because it only handles filtered positions, not the whole corpus text.

3.2 Details

Problem formulation. First, we formulate the *soft* pattern-matching problem by simply relaxing the classical *hard* (or exact) pattern-matching problem (Hakak et al., 2019) as follows:

Algorithm 1: Our soft pattern-matching algorithm.

Given : The corpus text $\boldsymbol{t} = t_1 t_2 \cdots t_N$ and its inverted index $I_v \subseteq \{1, \ldots, N\}$ over each vocabulary word $v \in \mathcal{V}$ such that $i \in I_v$ if and only if $v = t_i$.

Input : The phrase pattern $\boldsymbol{p} = p_1 p_2 \cdots p_n$ and the threshold $\alpha \in (0, 1]$.

Output : The set of soft match positions $M \subseteq \{1, \ldots, N\}$, such that $i \in M$ holds if and only if $p_k \approx_\alpha t_{i+k-1}$ holds over any $k = 1, \ldots, n$.

`// Step 1: Soften the pattern` $p_1 p_2 \cdots p_n$ `into the soft pattern` $S_1 S_2 \cdots S_n$

1 **for** $k \leftarrow 1 \ldots n$ **do**
2 $\quad S_k \leftarrow \varnothing$
3 $\quad$ **for** $v \in \mathcal{V}$ **do**
4 $\quad\quad$ **if** $v \approx_\alpha p_k$ **then** $S_k \leftarrow S_k \cup \{v\}$

`// Step 2-1: Get the soft inverted index` $\tilde{I}_k$ `by relaxing` I `with` S_k

5 **for** $k \leftarrow 1 \ldots n$ **do**
6 $\quad \tilde{I}_k \leftarrow \varnothing$
7 $\quad$ **for** $v \in S_k$ **do** $\tilde{I}_k \leftarrow \tilde{I}_k \cup I_v$

`// Step 2-2: Get the complete matching` M `by aggregating` $\tilde{I}_k$

8 $M \leftarrow \tilde{I}_1$
9 **for** $k \leftarrow 2 \ldots n$ **do** $M \leftarrow M \cap \left\{ i - (k-1) \;\middle|\; i \in \tilde{I}_k \right\}$
10 **return** M

The soft pattern-matching problem.
INPUT: The corpus text $\boldsymbol{t} = t_1 t_2 \cdots t_N \in \mathcal{V}^*$, the pattern $\boldsymbol{p} = p_1 p_2 \cdots p_n \in \mathcal{V}^*$, and the threshold $\alpha \in (0, 1]$
OUTPUT: The set of soft match positions M with respect to the soft equivalence $\approx_\alpha$, i.e., the set $M = \left\{ i \in \{1, \ldots, N\} \;\middle|\; \forall k \in \{1, \ldots, n\}.\, p_k \approx_\alpha t_{i+k-1} \right\}$

We define here the *soft equivalence* $v \approx_\alpha v'$ between words as $v \approx_\alpha v' \overset{\triangle}{\iff} \cos(E(v), E(v')) \geq \alpha$, to capture the similarity in terms of *word embeddings* E. Here, we let $\mathcal{V} = \{v_1, v_2, \ldots, v_L\}$ be the vocabulary and write $E(v) \in \mathbb{R}^D$ for the word embedding of a word $v \in \mathcal{V}$. Also, we write $\cos(\boldsymbol{e}, \boldsymbol{e}')$ for the cosine similarity $\cos(\boldsymbol{e}, \boldsymbol{e}') \triangleq \frac{\boldsymbol{e} \cdot \boldsymbol{e}'}{\|\boldsymbol{e}\| \|\boldsymbol{e}'\|}$ of word embeddings $\boldsymbol{e}, \boldsymbol{e}' \in \mathbb{R}^D$.

Preprocessing. Before processing queries, our algorithm preprocesses the whole corpus text to compute the *inverted index* I (Fig. 1a), following a standard technique in search engine indexing (Zobel & Moffat, 2006). The inverted index I maps each vocabulary word $v \in \mathcal{V}$ to the set of positions $I_v \subseteq \{1, \ldots, N\}$ that represents the occurences of the word v in the corpus text $\boldsymbol{t}$, that is, the set $I_v = \left\{ i \in \{1, \ldots, N\} \;\middle|\; v = t_i \right\}$.

Our algorithm. Algorithm 1 and Figures 1b to 1d outline our algorithm for soft pattern matching. Our algorithm works in the following two steps:

1. For each query, it performs *soft* computation over the *vocabulary*, *softening* the pattern $p_1 p_2 \cdots p_n$ into the *soft pattern* $S_1 S_2 \cdots S_n$ (lines 1 to 4 in Algorithm 1, Figure 1b). Here, S_k is the set of vocabulary words that *softly* matches each word p_k of the pattern in terms of the *soft* equivalence $\approx_\alpha$, i.e., $S_k = \left\{ v \in \mathcal{V} \;\middle|\; v \approx_\alpha p_k \right\}$.
2. Then, it performs *hard* computation that computes the *exact* set of positions M in the corpus text that matches the soft pattern $S_1 S_2 \cdots S_n$ by the following two sub-steps:
   2-1. Compute the *soft inverted index* $\tilde{I}$, which maps each pattern word p_k to the union $\tilde{I}_k = \bigcup_{v \in S_k} I_v$ of the exact inverted index I over S_k (lines 5 to 7, Fig. 1c). The set $\tilde{I}_k$ agrees with the set of positions that softly match the pattern word p_k, i.e., $\tilde{I}_k = \left\{ i \in \{1, \ldots, N\} \;\middle|\; t_i \approx_\alpha p_k \right\}$.
   2-2. Output the set of *soft matches* M by computing the shifting intersection $M = \bigcap_{k=1}^{n} \left\{ i - (k-1) \;\middle|\; i \in \tilde{I}_k \right\}$ over the soft inverted index $\tilde{I}$ (lines 8 to 9,

Fig. 1d). The resulting set precisely agrees with the expected set, i.e., $M = \{ i \in \{1, \ldots, N\} \mid \forall k \in \{1, \ldots, n\}. p_k \approx_\alpha t_{i+k-1} \}$.

We perform soft pattern matching using the index I. First, for each word p_k in $\boldsymbol{p}$, we construct the set $\tilde{I}_k$ indicating the positions of the words in $\boldsymbol{t}$ matching p_k (lines 1 to 4): we compare each word $v \in \mathcal{V}$ in the vocabulary with p_k (line 4); we add I_v to $\tilde{I}_k$ if we have $v \approx_\alpha p_k$ (line 7). Then, we construct the result M by aggregating each $\tilde{I}_k$ considering the position k of p_k in $\boldsymbol{p}$.

Inverted index vs. Suffix array. Our algorithm uses an inverted index rather than a suffix array (Yamamoto & Church, 2001), which is crucial. A suffix array manages *exact sequences* of characters (or tokens) in the corpus text, for which *softening* patterns cannot be performed efficiently. In contrast, entries of an inverted index I_v just have the occurrences of *each* word type v and can be relaxed for soft matching simply by merging the sets, as illustrated in Fig. 1c.

3.3 Formal analysis

How efficient is our algorithm? To answer this with theoretical guarantees, we analyze the time and space complexity of the algorithm. Overall, the high-level observation is that the corpus text size N affects the time and space required for preprocessing and soft matching only *linearly*.

Preprocessing. The time complexity for constructing the inverted index I is $\mathcal{O}(L \times N)$, consisting of $L \times N$ exact word comparisons, since each vocabulary word $v \in \mathcal{V}$ is compared with each word t_i in the corpus text $\boldsymbol{t}$. Notably, the constant factor here is typically very low, as only bitwise comparison is required. The total space required to store the inverted index I is only $\mathcal{O}(N + L)$, by simply storing the list of positions (of space $\mathcal{O}(|I_v|)$) for each vocabulary word $v \in \mathcal{V}$. This is because each position $i \in \{1, 2, \ldots, N\}$ in the text $\boldsymbol{t}$ occurs exactly once in I and hence $N = \sum_{v \in \mathcal{V}} |I_v|$. The space complexity is also $\mathcal{O}(N + L)$, since no extra space is required.

Soft matching. Step 1 for softening the pattern takes $\mathcal{O}(n \times L)$ in time, where the dominant part is $n \times L$ soft word comparisons (taking the cosine similarity of word embeddings) between each pattern word p_k and each vocabulary word $v \in \mathcal{V}$ (line 4). The space complexity is $\mathcal{O}(\sum_{k=1}^{n} |S_k|)$. Notably, the time and the space for Step 1 are independent of the corpus text size N. Step 2 for finding the set of soft matches takes $\mathcal{O}(K)$ in time and space, where K is the total size of the soft inverted index $K \triangleq \sum_{k=1}^{n} |\tilde{I}_k|$ (or roughly the total number of 'candidate' positions for matches). Remarkably, this is only linear to the corpus text size N.

4 Empirical evaluation

We address the following research questions in the experiments:

- **RQ1:** Does `SoftMatcha` scale to a billion-scale corpus, which is the typical size of training corpus for large causal LMs? (Section 4.2)
- **RQ2:** Does `SoftMatcha` perform as expected in typical scenarios in NLP and corpus linguistics? (Sections 4.2 and 4.3)

4.1 Implementation

We implemented the algorithm in Section 3 as a tool named `SoftMatcha`.[4] Our implementation adopts the following designs for efficiency. Firstly, we design our inverted index using a sparse matrix in a compressed sparse row (CSR) format to reduce memory consumption and enable efficient access to the index for each word, i.e., I_v. The index I is represented by a one-dimensional array, with the positions of each word contiguously allocated in memory. Secondly, we compile some time-consuming operations to native code using NUMBA (Lam et al., 2015). Specifically, line 4 in Algorithm 1 calculates word embedding similarities over $n \times L$ times, i.e., the time complexity is

[4] The word *Matcha* means powdered green tea in Japanese and is here a pun for 'matcher'. *Matcha softo* in Japanese means soft-serve ice cream of green tea flavor.

$\mathcal{O}(n \times L \times D)$, where D is the dimension of each word embedding. To speed up this calculation, we leverage the vectorized instruction set (SIMD) for parallel processing. In addition, finding the intersection of two large sets in line 9 in Algorithm 1 is computationally expensive. We employ just-in-time (JIT) compilation and parallel loops for efficient comparisons of elements. To ensure reproducibility, we release our source code on GitHub (Footnote 1) and the package on PyPI[5].

We provide a demo environment (Footnote 2) for users to explore the capabilities of our method and experience how diverse and linguistically natural the search results are and how quickly results can be obtained. The corpora used in experiments in Sections 4.2 and 4.3 are available. To prevent excessive output, for English and Japanese, only subsets of the corpora have been incorporated. Furthermore, search results are presented in smaller batches—to improve usability—with additional results available upon request, rather than all at once.

4.2 Case study in natural language processing — Billion-scale corpus search

In this section, we simulate scenarios where NLP researchers use `SoftMatcha` on billion-scale English and Japanese corpora. We focus on two types of evaluations: (i) quantitative evaluation — we assess the running time using large data comparable in size to those used for training standard large LMs; and (ii) qualitative evaluation — we examine the tool's effectiveness in detecting toxic instances within large LM training corpora.

Why English and Japanese? English is the dominant language in modern NLP and accounts for the largest portion of training data for large LMs (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023). In contrast, Japanese presents a typologically distinct language, with orthographic and structural features that differ significantly from English, such as: (i) character types (alphabetic vs. hiragana, katakana, and kanji); (ii) syntax (SVO vs. SOV, head-initial vs. head-final); and (iii) word segmentation (space-separated vs. no word separation) (Haspelmath et al., 2005). Furthermore, the substantial size of Japanese corpora (Wikipedia, 2024) makes it suitable for testing the scalability of our approach.

Setup. Corpora: we utilized the LLM-jp corpus v2.0 (LLM-jp et al., 2024), which contains organized collections of Wikipedia articles in both English (3.4B words) and Japanese (1.1B words). **Word embeddings:** we used GloVe `glove-wiki-gigaword-300` (Pennington et al., 2014) with a threshold $\alpha = 0.55$ for the English word embeddings, and fastText `facebook/fasttext-ja-vectors` (Grave et al., 2018) with a threshold $\alpha = 0.50$ for the Japanese word embeddings. **Baseline methods:** we compared the search results of `SoftMatcha` with those of exact matching, i.e., pattern matching with a standard inverted index, and dense vector search using `intfloat/multilingual-e5-large` model (Wang et al., 2024b). In dense vector search, we encoded each training instance[6] in the corpus and built the graph-based index using hierarchical navigable small worlds (HNSW) (Malkov & Yashunin, 2020) with the FAISS implementation (Douze et al., 2024) for the approximate nearest neighbor search. In HNSW, the number of edges was 32, the efConstruction was set to 40, and efSearch was configured as 16 (Malkov & Yashunin, 2020). We prepended the prefix "passage: " to each instance and "query: " to each query, following Wang et al. (2024b), and truncated any instances that exceeded 512 tokens. The text embeddings were calculated by averaging the contextualized token embeddings. **Computational environment:** we measured the running time on 152 core CPUs (Intel® Xeon® Platinum 8368 CPU @ 2.40GHz) and a 226 GiB main memory for the exact matching and `SoftMatcha`, and on 8 NVIDIA A100 GPUs for the dense vector search.

Running time. We measured the indexing time and search time in both the English and Japanese Wikipedia articles. Table 2 demonstrates that `SoftMatcha` is faster than dense vector search in both indexing and search time and takes the same indexing time as exact matching. Next, we investigated the relationship between the search speed and the corpus size. We constructed subsets of the English Wikipedia, whose sizes are $\{10^{-3}, 10^{-2}, 10^{-1}\}$ times that of the original corpus of 3.4B tokens, by randomly sampling from the original corpus, and measured the search time with each subset. Figure 2 shows the results. We observed that the search time increased only sublinearly,

[5] PyPI: https://pypi.org/project/softmatcha

[6] A training instance of LLM is a chunk of texts, which may consist of multiple sentences.

Table 2: Running time (sec) of indexing and search in the English and Japanese Wikipedia articles.

	En (3.4B words)		Ja (1.1B words)	
	Indexing	Search	Indexing	Search
Exact matching	685.8	0.005	242.5	0.022
SoftMatcha	685.8	0.098	242.5	0.055
Dense vector search	1036.5	0.389	320.4	0.283

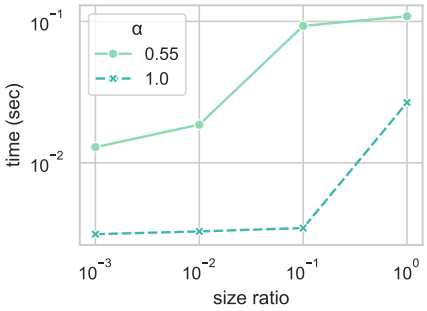


Figure 2: Running time on subsampled English Wikipedia corpora.

Table 3: Results of billion-scale corpus search in the English and Japanese Wikipedia articles.

	Exact matching	SoftMatcha	Dense vector search
Query: "*homemade bombs*" (2 tokens) in English Wikipedia (3.4B words)			
# Hits	107	1,473	n/a (depends on the top-k)
Match examples	*homemade bombs*	*homemade bombs*	Article: Survival Under Atomic Attack
		home-made grenades	Article: Mark 24 nuclear bomb
		homemade missiles	Article: List of common misconceptions
Query: "手製爆弾 (*homemade bombs*)" (2 tokens) in Japanese Wikipedia (1.1B words)			
# Hits	27	42	n/a (depends on the top-k)
Match examples	手製爆弾 (*homemade bombs*)	手製爆弾 (*homemade bombs*)	Article: 花火 (*fireworks*)
		手製手榴弾 (*home-made grenades*)	Article: 手持ち花火 (*consumer fireworks*)

not linearly, with respect to the increase in corpus size. We thus confirmed that our algorithm works effectively for billion-scale corpus searches.

Search results. Table 3 shows the results of the billion-scale corpus search. We queried "*homemade bombs*" in the English Wikipedia corpus and "手製爆弾 (*homemade bombs*)" in the Japanese Wikipedia corpus. In the dense vector search, we selected some retrieved examples from the top-10 search results. The table demonstrates that our SoftMatcha extends the exact matching. Note that the results of exact matching are a subset of the results of SoftMatcha. In Japanese, the dense vector search retrieved an article of "花火" (*fireworks*), which does not contain the contents related to the query. In contrast, SoftMatcha lists all contents that have the query pattern or its similar pattern. In addition, while dense vector search allows semantic similarity search, it cannot identify where the query pattern is located in a text. To summarize, a text that exactly contains the query pattern is not always retrieved in dense vector search, i.e., the recall is not always 100%, while exact matching and SoftMatcha can return all texts that contain the query pattern, and SoftMatcha also enables matching of semantically similar patterns.

4.3 Case study in corpus linguistics — Retrieving Latin examples

Why Latin? Latin is a morphologically complex fusional language where a lemma (dictionary form) may exhibit numerous different word forms depending on the morphological features (*e.g.*, voice, mood, tense, aspect, person, and number for verbals; gender, number, and case for nominals) that the word bears. For example, a transitive verb may conjugate to more than 100 different finite verb forms. This morphological complexity makes it harder to search through a corpus by exact matching. Furthermore, due to Latin's philological significance and the vast body of accumulated literature, there has been a persistent demand for advanced search tools to facilitate corpus analysis (Bamman & Smith, 2012). For these reasons, Latin is a suitable touchstone to test the utility of our proposed soft pattern matcher, particularly for humanities researchers and language learners.

Table 4: Latin match examples by `SoftMatcha` with queries *factus est* 'he/it was done/finished/made' and *equus est* 'it is a horse'. The top row of an interlinear gloss represents words (and the cosine similarity between the matched word and its corresponding query word), the middle row their morphological analysis, and the bottom row the free translation. Morphological matches are highlighted in pink, semantics matches in blue, and exact matches in green.

Query	
factus	***est***
do.PASS.PF.PTCP-M.NOM.SG	be.IND.PRS.3SG
'*he/it is done/finished/made*'	
Matches	
0.56	0.56
facta	***sunt***
do.PASS.PF.PTCP-N.NOM.PL	be.IND.PRS.3PL
'*they are done*' or '*they are facts*'	
0.53	0.58
mortuus	***esset***
die.ACT.PF.PTCP-M.NOM.SG	be.SUB.IMPF.3SG
'*he/it was dead*'	
0.65	0.65
creatus	***erat***
create.PASS.PF.PTCP-M.NOM.SG	be.IND.IMPF.3SG
'*he/it was created*'	

Query	
equus	***est***
horse.M.NOM.SG	be.IND.PRS.3SG
'*it is a horse*'	
Matches	
0.48	1.00
bos	***est***
cow.F.NOM.SG	be.IND.PRS.3SG
'*it is a cow.*'	
0.44	0.63
currus	***fuit***
chariot.M.NOM.SG	be.IND.PF.3SG
'*it was a chariot*'	
0.49	0.58
Minotaurus	***esset***
Minotaur.M.NOM.SG	be.SUB.IMPF.3SG
'*it would be the Minotaur*'	

Setup. Corpora: we used two corpora: one from the Perseus Project (Crane, 2023) (5M tokens) and the Augustinian Sermon Parallelism (ASP) Dataset (Bothwell et al., 2023) (0.1M tokens). **Word embeddings:** we used the pre-trained fastText embeddings `facebook/fasttext-la-vectors` (Grave et al., 2018).

Search results. Table 4 shows the returned matches with the queries *factus est* ('it/he is done' or 'it/he is made') and *equus est* ('it is a horse'). It is evident that `SoftMatcha` effectively links the queries to semantically similar words, as seen in *mortuus* 'dead' and *creatus* 'created' matched with the query *factus* 'done, finished, made', and *bos* 'cow', *currus* 'chariot', and *Minotaurus* 'Minotaur' with *equus* 'horse'. Interestingly, the tool is also able to catch different word forms with different morphological features while sharing the same lemma, which are highlighted in pink. For example, although the Latin copula verb in the query *est* 'is' exhibits highly irregular conjugation patterns, the matches successfully include their inflected forms such as *sunt*, *esset*, *erat*, and *fuit*. Furthermore, the matches *mortuus* and *creatus* are not only semantically similar to the query word *factus* but also have morphological features in common (perfect, participle, masculine, nominative, and singular).

5 Conclusion

In this paper, we propose a new pattern-matching algorithm that can flexibly handle the orthographic diversity of natural languages while also performing efficiently on large-scale corpora. Our algorithm, combining word embeddings and inverted indexing, achieves inference speed independent of corpus size. We have also developed and released a simple-to-use web demo for researchers and practitioners. For quantitative evaluation, we confirm that the developed tool can enumerate all search results within one second on billion-scale corpora, a typical scenario in training large LMs. For qualitative evaluation, we assess the tool in typical scenarios in NLP (detecting harmful examples from large-scale Japanese and English Wikipedia corpora) and computational linguistics (example retrieval from Latin, a language with highly diverse inflections), observing that our method can retrieve semantically similar examples that hard pattern matchers would miss.

ACKNOWLEDGMENTS

We conducted experiments using the language and computational resources of LLM-jp (https://llm-jp.nii.ac.jp). This research was supported in part by JSPS KAKENHI JP24KJ0133, JSPS KAKENHI JP23K24910, JST ACT-X JPMJAX200S, JST ACT-X JPMJAX200U, JST FOREST JPMJFR2331, JST PREST JPMJPR22CA, and JST CREST JPMJCR2012.

ETHICS STATEMENT

The corpora used for training large LMs, composed of web corpora, including Common Crawl (Common Crawl, 2024) and digitized book data such as (Zhu et al., 2015), can be said to encompass a substantial portion of humanity's linguistic knowledge. Inevitably, these corpora include a significant amount of ethically problematic content. This includes personal information (Subramani et al., 2023), as well as information that could be "beneficial" for violence and terrorism (or in more conventional terms, harmful information) (Albalak et al., 2024). Moreover, as these corpora continue to grow exponentially and our languages possess numerous paraphrases, comprehensively identifying such harmful learning resources is far from a simple task. Our research aims to substantially simplify this crucial activity for human ethics: identifying harmful learning instances within vast linguistic resources Section 4.2. We hope that, by leveraging our tool, NLP researchers and developers will contribute to providing LLMs as a safe and reliable social infrastructure.

REPRODUCIBILITY STATEMENT

Our source code is available on GitHub (Footnote 1), and we released an installable package on PyPI (Footnote 5). In all experiments, we used open datasets. Further details on the embeddings, corpora, computational environment, and protocols used in experiments are explained in Section 4.

REFERENCES

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, Colin Raffel, Shiyu Chang, Tatsunori Hashimoto, and William Yang Wang. A Survey on Data Selection for Language Models. *arXiv [cs.CL]*, 26 February 2024. URL http://arxiv.org/abs/2402.16827.

Laurence Anthony. A critical look at software tools in corpus linguistics. *Linguistic Research*, 30 (2):141–161, 2013.

Orlando Ayala and Patrice Bechard. Reducing hallucination in structured outputs via retrieval-augmented generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pp. 228–238, Stroudsburg, PA, USA, 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.naacl-industry.19.pdf.

John Bambenek and Agnieszka Klus. *Grep Pocket Reference: A Quick Pocket Reference for a Utility Every UNIX User Needs*. O'Reilly Media, 2009.

David Bamman and David Smith. Extracting two thousand years of latin from a million book library. *J. Comput. Cult. Herit.*, 5(1), April 2012. ISSN 1556-4673. doi: 10.1145/2160165.2160167. URL https://doi.org/10.1145/2160165.2160167.

Ezio Bartocci, Yliès Falcone, Adrian Francalanza, and Giles Reger. Introduction to runtime verification. In *Lectures on Runtime Verification*, volume 10457 of *Lecture Notes in Computer Science*, pp. 1–33. Springer, 2018.

Michael Bendersky, Honglei Zhuang, Ji Ma, Shuguang Han, Keith Hall, and Ryan McDonald. Rrf102: Meeting the trec-covid challenge with a 100+ runs ensemble, 2020. URL https://arxiv.org/abs/2010.00200.

Douglas Biber. Corpus-Based and Corpus-Driven Analyses of Language Variation and Use. In *The Oxford Handbook of Linguistic Analysis*. Oxford Academic, 2nd edition, 2015. doi: 10.1093/oxfordhb/9780199677078.013.0008.

Stella Biderman, USVSN Sai Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. Emergent and Predictable Memorization in Large Language Models, 2023a. URL https://arxiv.org/abs/2304.11158.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. *arXiv [cs.CL]*, 3 April 2023b. URL http://arxiv.org/abs/2304.01373.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguist.*, 5:135–146, December 2017. URL https://aclanthology.org/Q17-1010.pdf.

Stephen Bothwell, Justin DeBenedetto, Theresa Crnkovich, Hildegund Müller, and David Chiang. Introducing Rhetorical Parallelism Detection: A New Task with Datasets, Metrics, and Baselines. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5007–5039, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.305. URL https://aclanthology.org/2023.emnlp-main.305.

Robert S. Boyer and J Strother Moore. A Fast String Searching Algorithm. *Commun. ACM*, 20 (10):762–772, 1977. doi: 10.1145/359842.359859. URL https://doi.org/10.1145/359842.359859.

Leonid Boytsov. Indexing methods for approximate dictionary searching: Comparative analysis. *ACM J. Exp. Algorithmics*, 16(1), 2011.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Michael Burrows, D J Wheeler D I G I T A L, Robert W. Taylor, David J. Wheeler, and David Wheeler. A block-sorting lossless data compression algorithm. 1994. URL https://api.semanticscholar.org/CorpusID:2167441.

Julius Caesar. *Commentarii de bello gallico. With explanatory notes, lexicon, maps, indexes, etc.* Eldredge & Brother, Philadelphia, 1882.

Yi-Pei Chen, Noriki Nishida, Hideki Nakayama, and Yuji Matsumoto. Recent Trends in Personalized Dialogue Generation: A Review of Datasets, Methodologies, and Evaluations. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 13650–13665, 2024. URL https://aclanthology.org/2024.lrec-main.1192.pdf.

Yuanyuan Chen, Boyang Li, Han Yu, Pengcheng Wu, and Chunyan Miao. HyDRA: Hypergradient Data Relevance Analysis for Interpreting Deep Neural Networks. *Proc. Conf. AAAI Artif. Intell.*, 35(8):7081–7089, 18 May 2021. URL https://cdn.aaai.org/ojs/16871/16871-13-20365-1-2-20210518.pdf.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh,

Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling language modeling with Pathways. *arXiv [cs.CL]*, 5 April 2022. URL http://arxiv.org/abs/2204.02311.

Common Crawl. Common Crawl - Open Repository of Web Crawl Data, 2024. URL https://commoncrawl.org/.

Matthew Coole, Paul Rayson, and John Mariani. LexiDB: Patterns & Methods for Corpus Linguistic Database Management. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 3128–3135, 2020. URL https://aclanthology.org/2020.lrec-1.383.pdf.

Gregory Crane. The Perseus Digital Library and the future of libraries. *International Journal on Digital Libraries*, 2023. doi: 10.1007/s00799-022-00333-2.

Chris Culy and Verena Lyding. Double Tree: An Advanced KWIC Visualization for Expert Users. In *2010 14th International Conference Information Visualisation*, pp. 98–103, 2010. doi: 10.1109/IV.2010.24.

Daniel De Freitas. Announcing our Series A and our new AI model, C1.2. https://blog.character.ai/character-ai/, 23 March 2023. Accessed: 2024-10-1.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4171–4186, Stroudsburg, PA, USA, 2019a. Association for Computational Linguistics. URL https://aclanthology.org/N19-1423.pdf.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019b. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Jan Don. *Morphological theory and the morphology of English*. Edinburgh University Press, 2014. doi: 10.1515/9780748645145.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library, 2024. URL https://arxiv.org/abs/2401.08281.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah

Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong,

Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The Llama 3 herd of models. *arXiv [cs.AI]*, 31 July 2024. URL http://arxiv.org/abs/2407.21783.

David Embick. *The morpheme: A theoretical introduction*. De Gruyter Mouton, Berlin, München, Boston, 2015. doi: 10.1515/9781501502569.

Paolo Ferragina and Giovanni Manzini. Indexing compressed text. *J. ACM*, 52(4):552–581, July 2005. ISSN 0004-5411. doi: 10.1145/1082036.1082039. URL https://doi.org/10.1145/1082036.1082039.

Andrew Gallant. ripgrep: ripgrep recursively searches directories for a regex pattern while respecting your gitignore, 2024. URL https://github.com/BurntSushi/ripgrep.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The Paraphrase Database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 758–764, 2013. URL https://aclanthology.org/N13-1092.pdf.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800GB dataset of diverse text for language modeling. *arXiv [cs.CL]*, 31 December 2020. URL http://arxiv.org/abs/2101.00027.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv [cs.CL]*, 18 December 2023. URL http://arxiv.org/abs/2312.10997.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3356–3369, Stroudsburg, PA, USA, November 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.findings-emnlp.301.pdf.

Adrien Grand, Robert Muir, Jim Ferenczi, and Jimmy Lin. From MAXSCORE to Block-Max Wand: The Story of How Lucene Significantly Improved Query Evaluation Performance. In *ECIR (2)*, volume 12036 of *Lecture Notes in Computer Science*, pp. 20–27. Springer, 2020.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning Word Vectors for 157 Languages, 2018. URL https://arxiv.org/abs/1802.06893.

Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. A Survey on Automated Fact-Checking. *Transactions of the Association for Computational Linguistics*, 10:178–206, 2022. doi: 10.1162/tacl_a_00454. URL https://aclanthology.org/2022.tacl-1.11.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-Augmented Language Model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Le Quan Ha, Philip Hanna, Ji Ming, and F. J. Smith. Extending Zipf's law to n-grams for large corpora. *Artificial Intelligence Review*, 2009. doi: 10.1007/s10462-009-9135-4.

Saqib Hakak, Amirrudin Kamsin, Palaiahnakote Shivakumara, Gulshan Amin Gilkar, Wazir Zada Khan, and Muhammad Imran. Exact String Matching Algorithms: Survey, Issues, and Future Research Directions. *IEEE Access*, 7:69614–69637, 2019. doi: 10.1109/ACCESS.2019.2914071. URL https://doi.org/10.1109/ACCESS.2019.2914071.

Martin Haspelmath, Martin S Dryer, David Gil, and Bernard Comrie. *The World Atlas of Language Structures*. Oxford University Press, London, England, 21 July 2005.

H. S. Heap. Information retrieval: Computational and theoretical aspects. *Libr. Q.*, 50(1):153–154, January 1980. URL http://dx.doi.org/10.1086/629887.

Susan Hockey and Jeremy Martin. The Oxford Concordance Program version 2. *Literary and Linguistic Computing*, 2(2):125–131, January 1987. doi: 10.1093/llc/2.2.125.

Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. Are Large Pre-Trained Language Models Leaking Your Personal Information? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 2038–2047, December 2022. URL https://aclanthology.org/2022.findings-emnlp.148.pdf.

Daphne Ippolito, Florian Tramer, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher Choquette Choo, and Nicholas Carlini. Preventing generation of verbatim memorization in language models gives a false sense of privacy. In C. Maria Keet, Hung-Yi Lee, and Sina Zarrieß (eds.), *Proceedings of the 16th International Natural Language Generation Conference*, pp. 28–53, Prague, Czechia, September 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.inlg-main.3. URL https://aclanthology.org/2023.inlg-main.3.

Masaru Isonuma and Ivan Titov. Unlearning Traces the Influential Training Data of Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6312–6325, Stroudsburg, PA, USA, 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.343.pdf.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised Dense Information Retrieval with Contrastive Learning, 2022. URL https://arxiv.org/abs/2112.09118.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot Learning with Retrieval Augmented Language Models. *J. Mach. Learn. Res.*, 24(1), March 2024. ISSN 1532-4435.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

Kim Ebensgaard Jensen. Linguistics in digital humanities: (computational) corpus linguistics. *MedieKultur: Journal of Media and Communication Research*, 30(57), 2014. doi: 10.7146/mediekultur.v30i57.15968.

Leonard B Jung, Jonas A Gudera, Tim L T Wiegand, Simeon Allmendinger, Konstantinos Dimitriadis, and Inga K Koerte. ChatGPT Passes German State Examination in Medicine With Picture Questions Omitted. *Dtsch. Arztebl. Int.*, 120(21):373–374, 30 May 2023. URL http://dx.doi.org/10.3238/arztebl.m2023.0113.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd edition, 2024. URL https://web.stanford.edu/~jurafsky/slp3/. Online manuscript released August 20, 2024.

Richard M. Karp and Michael O. Rabin. Efficient Randomized Pattern-Matching Algorithms. *IBM J. Res. Dev.*, 31(2):249–260, 1987.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://www.aclweb.org/anthology/2020.emnlp-main.550.

Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo Arredondo. GPT-4 Passes the Bar Exam. *Philos. Trans. A Math. Phys. Eng. Sci.*, 382(2270):20230254, 15 April 2024. URL http://dx.doi.org/10.1098/rsta.2023.0254.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through Memorization: Nearest Neighbor Language Models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HklBjCEKvH.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, 25 July 2020. ACM. URL https://dl.acm.org/doi/10.1145/3397271.3401075.

Donald E. Knuth, James H. Morris Jr., and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6(2):323–350, 1977.

Tatsuru Kobayashi and Kumiko Tanaka-Ishii. Taylor's law for Human Linguistic Sequences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1138–1148, Stroudsburg, PA, USA, 2018. Association for Computational Linguistics. URL https://aclanthology.org/P18-1105.pdf.

Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. *arXiv [stat.ML]*, 14 March 2017. URL http://arxiv.org/abs/1703.04730.

Sachin Kumar, Vidhisha Balachandran, Lucille Njoo, Antonios Anastasopoulos, and Yulia Tsvetkov. Language Generation Models Can Cause Harm: So What Can We Do About It? An Actionable Survey. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3299–3321, Stroudsburg, PA, USA, 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.eacl-main.241.pdf.

Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: a LLVM-based Python JIT compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450340052. doi: 10.1145/2833157.2833162. URL https://doi.org/10.1145/2833157.2833162.

B Langmead, C Trapnell, M Pop, and S Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*, 10:R25, 2009. doi: 10.1186/gb-2009-10-3-r25.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-Tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv [cs.CL]*, 22 May 2020. URL http://arxiv.org/abs/2005.11401.

Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. Multi-step Jailbreaking Privacy Attacks on ChatGPT. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 4138–4153, Stroudsburg, PA, USA, December 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.findings-emnlp.272.pdf.

Jiacheng Liu, Sewon Min, Luke Zettlemoyer, Yejin Choi, and Hannaneh Hajishirzi. Infini-gram: Scaling Unbounded n-gram Language Models to a Trillion Tokens. *arXiv [cs.CL]*, 30 January 2024. URL http://arxiv.org/abs/2401.17377.

LLM-jp, :, Akiko Aizawa, Eiji Aramaki, Bowen Chen, Fei Cheng, Hiroyuki Deguchi, Rintaro Enomoto, Kazuki Fujii, Kensuke Fukumoto, Takuya Fukushima, Namgi Han, Yuto Harada, Chikara Hashimoto, Tatsuya Hiraoka, Shohei Hisada, Sosuke Hosokawa, Lu Jie, Keisuke Kamata, Teruhito Kanazawa, Hiroki Kanezashi, Hiroshi Kataoka, Satoru Katsumata, Daisuke Kawahara, Seiya Kawano, Atsushi Keyaki, Keisuke Kiryu, Hirokazu Kiyomaru, Takashi Kodama, Takahiro Kubo, Yohei Kuga, Ryoma Kumon, Shuhei Kurita, Sadao Kurohashi, Conglong Li, Taiki Maekawa, Hiroshi Matsuda, Yusuke Miyao, Kentaro Mizuki, Sakae Mizuki, Yugo Murawaki, Ryo Nakamura, Taishi Nakamura, Kouta Nakayama, Tomoka Nakazato, Takuro Niitsuma, Jiro Nishitoba, Yusuke Oda, Hayato Ogawa, Takumi Okamoto, Naoaki Okazaki, Yohei Oseki, Shintaro Ozaki, Koki Ryu, Rafal Rzepka, Keisuke Sakaguchi, Shota Sasaki, Satoshi Sekine, Kohei Suda, Saku Sugawara, Issa Sugiura, Hiroaki Sugiyama, Hisami Suzuki, Jun Suzuki, Toyotaro Suzumura, Kensuke Tachibana, Yu Takagi, Kyosuke Takami, Koichi Takeda, Masashi Takeshita, Masahiro Tanaka, Kenjiro Taura, Arseny Tolmachev, Nobuhiro Ueda, Zhen Wan, Shuntaro Yada, Sakiko Yahata, Yuya Yamamoto, Yusuke Yamauchi, Hitomi Yanaka, Rio Yokota, and Koichiro Yoshino. LLM-jp: A Cross-organizational Project for the Research and Development of Fully Open Japanese LLMs, 2024. URL https://arxiv.org/abs/2407.03963.

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Analyzing Leakage of Personally Identifiable Information in Language Models. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2023. URL http://dx.doi.org/10.1109/sp46215.2023.10179300.

Yu A. Malkov and D. A. Yashunin. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42 (4):824–836, April 2020. ISSN 0162-8828. doi: 10.1109/TPAMI.2018.2889473. URL https://doi.org/10.1109/TPAMI.2018.2889473.

Tony McEnery and Andrew Hardie. *Corpus Linguistics: Method, Theory and Practice*. Cambridge University Press, 2011.

Tony McEnery and Andrew Wilson. *Corpus Linguistics*. Edinburgh University Press, Edinburgh, 2001. doi: 10.1515/9781474470865.

Barbara Mcgillivray, Thierry Poibeau, and Pablo Ruiz. Digital Humanities and Natural Language Processing: "Je t'aime... Moi non plus". *Digital Humanities Quarterly*, 14(2), 2020. doi: 10.17863/CAM.55816.

Kathleen R McKeown. Paraphrasing Using Given and New Information in a Question-Answer System. In *Proceedings of the 17th annual meeting on Association for Computational Linguistics* -, pp. 67–72, Morristown, NJ, USA, 1979. Association for Computational Linguistics. URL https://aclanthology.org/P79-1016.pdf.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv [cs.CL]*, 16 January 2013. URL http://arxiv.org/abs/1301.3781.

D. Lee Miller. Wordllama: Recycled token embeddings from large language models, 2024. URL https://github.com/dleemiller/wordllama.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 2014–2037, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.148. URL https://aclanthology.org/2023.eacl-main.148/.

Gonzalo Navarro. A Guided Tour to Approximate String Matching. *ACM Comput. Surv.*, 33(1): 31–88, 2001.

OpenAI. Introducing OpenAI o1. https://openai.com/index/introducing-openai-o1-preview/, 12 September 2024. Accessed: 2024-10-1.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating Training Data Influence by Tracing Gradient Descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/e6385d39ec9394f2f3a354d9d2b88eec-Paper.pdf.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv [cs.LG]*, 23 October 2019. URL http://arxiv.org/abs/1910.10683.

Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju-Yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Perrin, Sébastien M R Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving Open Language Models at a Practical Size. *arXiv [cs.CL]*, 31 July 2024. URL http://arxiv.org/abs/2408.00118.

Kirk Roberts, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, Kyle Lo, Ian Soboroff, Ellen Voorhees, Lucy Lu Wang, and William R. Hersh. Searching for scientific evidence in a pandemic:

An overview of trec-covid. *J. of Biomedical Informatics*, 121(C), September 2021. ISSN 1532-0464. doi: 10.1016/j.jbi.2021.103865. URL https://doi.org/10.1016/j.jbi.2021.103865.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009. ISSN 1554-0669. doi: 10.1561/1500000019. URL https://doi.org/10.1561/1500000019.

Marek Rogozinski and Rafal Kuc. *Elasticsearch Server, 3rd edition*. Packt Publishing, 2016.

Timo Schick, Sahana Udupa, and Hinrich Schütze. Self-Diagnosis and Self-Debiasing: A Proposal for Reducing Corpus-Based Bias in NLP. *Trans. Assoc. Comput. Linguist.*, 9:1408–1424, 17 December 2021. URL https://aclanthology.org/2021.tacl-1.84.pdf.

Sarah Schulz. *The Taming of the Shrew: Non-Standard Text Processing in the Digital Humanities*. PhD thesis, University of Stuttgart, 2018.

Martin Schweinberger. Concordancing with R. https://ladal.edu.au/kwics.html, 2024.

Satoshi Sekine. A Linguistic Knowledge Discovery Tool: Very Large Ngram Database Search with Arbitrary Wildcards. In *Coling 2008: Companion volume: Demonstrations*, pp. 181–184, 2008. URL https://aclanthology.org/C08-3010.pdf.

Satoshi Sekine and Kapil Dalwani. Ngram Search Engine with Patterns Combining Token, POS, Chunk and NE Information. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, 2010. URL http://www.lrec-conf.org/proceedings/lrec2010/pdf/158_Paper.pdf.

Frank Smadja. Retrieving Collocations from Text: Xtract, 1993. URL https://aclanthology.org/J93-1007.pdf.

Nishant Subramani, Sasha Luccioni, Jesse Dodge, and Margaret Mitchell. Detecting Personal Information in Training Corpora: an Analysis. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pp. 208–220, Stroudsburg, PA, USA, 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.trustnlp-1.18.pdf.

TEI Consortium. TEI P5: Guidelines for electronic text encoding and interchange, 2023. URL https://www.tei-c.org/Guidelines/P5.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv [cs.CL]*, 18 July 2023. URL http://arxiv.org/abs/2307.09288.

Janneke Van Der Zwaan, Wouter Smink, Anneke Sools, Gerben Westerhof, Bernard Veldkamp, and Sytske Wiegersma. Flexible NLP Pipelines for Digital Humanities Research. In *4th Digital Humanities Benelux Conference*, Utrecht, Netherlands, 2017.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text Embeddings by Weakly-Supervised Contrastive Pre-training, 2024a. URL https://arxiv.org/abs/2212.03533.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual e5 text embeddings: A technical report, 2024b. URL https://arxiv.org/abs/2402.05672.

Yuxia Wang, Revanth Gangi Reddy, Zain Muhammad Mujahid, Arnav Arora, Aleksandr Rubashevskii, Jiahui Geng, Osama Mohammed Afzal, Liangming Pan, Nadav Borenstein, Aditya Pillai, Isabelle Augenstein, Iryna Gurevych, and Preslav Nakov. Factcheck-bench: Fine-grained evaluation benchmark for automatic fact-checkers, 2024c. URL https://arxiv.org/abs/2311.09000.

Wikipedia. Wikipedia:Multilingual statistics, 2024. URL https://en.wikipedia.org/wiki/Wikipedia:Multilingual_statistics.

Sam Witteveen and Martin Andrews. Paraphrasing with Large Language Models. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pp. 215–220, Stroudsburg, PA, USA, November 2019. Association for Computational Linguistics. URL https://aclanthology.org/D19-5623.pdf.

Sun Wu and Udi Manber. Fast Text Searching Allowing Errors. *Commun. ACM*, 35(10):83–91, 1992a. doi: 10.1145/135239.135244. URL https://doi.org/10.1145/135239.135244.

Sun Wu and Udi Manber. Agrep – A Fast Approximate Pattern-Matching Tool. In *1992 Winter USENIX Conference*, 1992b.

Mikio Yamamoto and Kenneth W Church. Using Suffix Arrays to Compute Term Frequency and Document Frequency for All Substrings in a Corpus. *Comput. Linguist. Assoc. Comput. Linguist.*, 27(1):1–30, March 2001. URL https://aclanthology.org/J01-1001.pdf.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. *arXiv [cs.CV]*, 22 June 2015. URL http://arxiv.org/abs/1506.06724.

George Kingsley Zipf. Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology. *Am. J. Psychol.*, 64(1):149, January 1951. URL http://dx.doi.org/10.2307/1418618.

Justin Zobel and Alistair Moffat. Inverted Files for Text Search Engines. *ACM Comput. Surv.*, 38(2):6, 2006. doi: 10.1145/1132956.1132959. URL https://doi.org/10.1145/1132956.1132959.

A Other examples of soft matches in Latin

Table 5 shows several additional examples of Latin soft matches by `SoftMatcha`.

B List of Glossing Abbreviations

Table 6 lists the glossing abbreviations used in this paper.

C Web Interface

Figures 3 to 5 shows screenshots of our demo tool.

Table 5: Additional examples of soft matches in Latin and their scores returned by `SoftMatcha`. The top row of an interlinear gloss represents words (and the cosine similarity between the word and its corresponding query word), the middle row their morphological analysis, and the bottom row the free translation. Morphological matches are highlighted in pink, semantic matches in blue, and exact matches in green.

Query

non	***potest***
not	can.IND.PRS.3SG

'*he/she/it cannot*'

Matches

1.00	0.59
non	***possit***
not	can.SUB.PRS.3SG

'*he/she/it cannot*'

0.53	0.52
nec	***posse***
nor	can.INF.PRS

'*not to be able*'

Query

homo	***sapiens***
human.M.NOM.SG	wise.M.NOM.SG

'*a wise human*'

Matches

1.00	0.39
homo	***honestus***
human.M.NOM.SG	noble-M.NOM.SG

'*an honorable human*'

0.47	0.40
vir	***sincerus***
man.M.NOM.SG	pure-M.NOM.SG

'*a pure man*'

Query[7]

post	***meridiem***
after	noon.M.ACC.SG

'*after noon*'

Matches

0.68	1.00
ante	***meridiem***
before	noon.M.ACC.SG

'*before noon*'

0.51	0.52
contra	***septentrionem***
against	Ursa.Major.F.ACC.SG

'*against Ursa Major*', i.e., '*facing north*'

Query[8]

bellum	***gallicum***
war.N-NOM.SG	gallic-N.NOM.SG

'*the Gallic war*'

Matches

1.00	0.44
bellum	***etruscum***
war.N-NOM.SG	Etruscan-N.NOM.SG

'*the Etruscan war*'

0.49	0.45
contra	***Caesarem***
against	Caesar.M-ACC.SG

'*against Caesar*'

Query[9]

quo	***vadis***
where	go.IND.PRS.2SG

'*where do you go*'

Matches

0.42	1.00
ibi	***vadis***
there	go.IND.PRS.2SG

'*you go there*'

0.50	0.48
autem	***vocaris***
but	summon-PASS.IND.PRS.2SG

'*but you are summoned*'

Query[10]

quod	***erat***	***demonstrandum***
which	be.IND.IMPF.3SG	show.GER-N.NOM.SG

'*which was to be shown*'

Matches

0.46	1.00	0.41
haec	***erat***	***forma***
this.F.NOM.SG	be.IND.IMPF.3SG	form.F.NOM.SG

'*this was the form*'

1.00	0.54	0.47
quod	***postea***	***accipiamus***
which	afterwards	accept.ACT.SUB.PRS.1PL

'*which we shall accept later*'

Table 6: List of the gloss abbreviations used in this paper.

Gloss	Meaning
1, 2, 3	1st, 2nd, 3rd person, respectively
ACC	accusative
ACT	active voice
F	feminine (gender)
FUT	future
GER	gerundive
IMPF	imperfective
IND	indicative mood
M	masculinie (gender)
N	neuter (gender)
NOM	nominative case
PASS	passive voice
PF	perfective
PL	plural
PRS	present tense
PTCP	participle
SG	singular
SUB	subjunctive mood

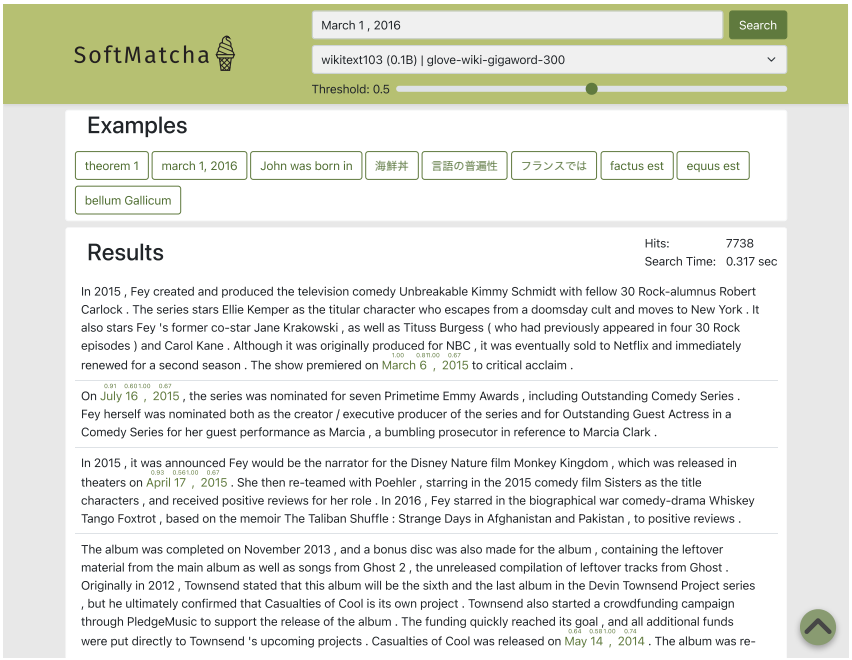


Figure 3: The screenshot of our demo. Given the query “March 1, 2016”, it returns lines including softly-matched patterns such as “July 16, 2015.”

D Effects of varying thresholds on search time and results

We inspected the relationship between the value of α , search time, and the number of matched entries for the query “homemade bomb”. We varied the threshold parameter, i.e., α from

[7] p.m.
[8] book by Caesar (1882)
[9] Bible, John 13:36
[10] Q.E.D.

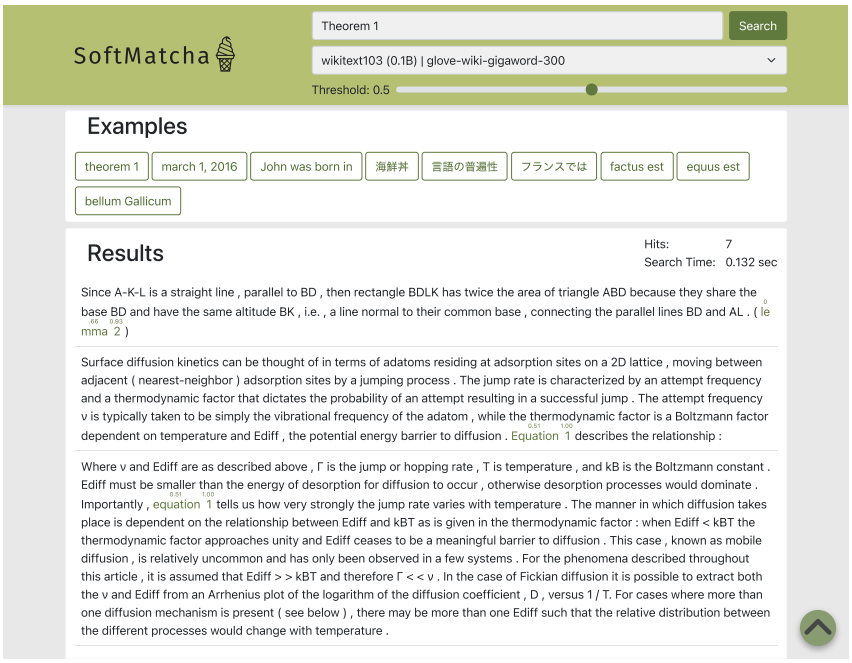


Figure 4: The screenshot of our demo. Given the query "Theorem 1", it returns lines including softly-matched patterns such as "lemma 2."

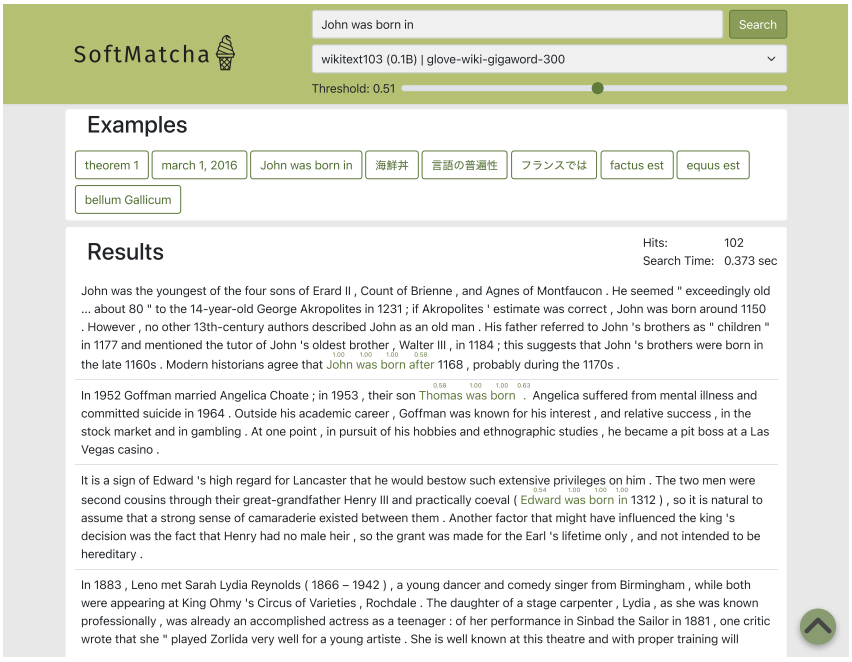


Figure 5: The screenshot of our demo. Given the query "John was born in," it returns lines including softly-matched patterns such as "Edward was born in."

$\{0.1, 0.2, \ldots, 1.0\}$, in the English Wikipedia search. The search time and the number of matched entries for each α are shown in Table 7. As you can see, both the search time and the number of matched entries vary significantly depending on the threshold. The results of $\alpha \leq 0.2$ are noteworthy. From the result, we can observe that the search time of the tool is too long to be used for real-time use if the alpha is set below 0.2. However, we remark that the threshold as low as 0.2 is not

Table 7: The search time (seconds) and the number of matched entries for each α.

α	seconds	# of hits
0.1	1213.503	504,955,515
0.2	55.854	19,899,652
0.3	1.105	183,202
0.4	0.258	34,766
0.5	0.079	1,839
0.6	0.025	381
0.7	0.028	259
0.8	0.027	259
0.9	0.028	107
1.0	0.005	107

chosen in practice since the search result includes too many nonsensical phrases. For example, the results for $\alpha = 0.2$ included "mixing test" and "authentic scale", which are not related to the query.

E Additional discussions

E.1 Out-of-vocabulary problem

Our current implementation does not handle out-of-vocabulary (OOV) words; it ignores unknown words. Handling OOV words appropriately is deferred to future work. We plan to (1) use fasttext to fallback unknown words to character embeddings and (2) extend our algorithm to handle subword units to deal with the OOV problem.

E.2 Static embeddings vs. dynamic embeddings

We used static embeddings for all experiments, but dynamic embeddings could also be used.

The simple one is to use the embedding layer of contextualized models such as BERT (Devlin et al., 2019b), taking an approach like WordLlama (Miller, 2024). We conducted a follow-up experiment using the embedding layer of BERT[11] for the search in the English Wikipedia with a threshold of $\alpha = 0.6$. From the experiment, our `SoftMatcha` with the embedding layer of BERT newly retrieved "makeshift bomb" and "improvised bomb" etc. with a query of "homemade bombs". It took less than one second, just as fast as when using the GloVe embedding. These new matches may seem a bit counterintuitive, but they suggest that we can possibly obtain deep semantic matches using more dynamic embeddings.

The other more radical one is to actually use contextualized token embeddings instead of mere words for the search keys. Although the contextualized embeddings can be arbitrary real vectors and are not suitable for indexing as is, we can approximate them to a finite number of embeddings by vector quantization or other methods to apply our algorithm. The context information of the query can be strengthened by adding some extra phrases before and after the query.

E.3 Omission, insertion, and swapping

The current `SoftMatcha` does not treat word omission and insertion, e.g., "the jazz musician" does not match "a fantastic jazz musician". That said, omission and insertion could be handled by modifying Step 2-2 (technically, how to take intersections in line 9, Algorithm 1) using wildcards. Wildcards can skip matching at any position, and is easy to implement by just adjusting the shift width of Step 2-2 in Algorithm 1. This extension allows us to represent various similar patterns with a single simple query.

In addition, it is possible to extend the algorithm to allow the flexible order of words in the query by considering every permutation of the query pattern. This can be quite efficient, because only Step

[11] `google-bert/bert-base-uncased`

Table 8: The results of the information retrieval task on the TREC-COVID dataset.

Method	P@20	R@1000	NDCG@20
BM25	39.5	22.2	34.6
Soft-BM25 ($\alpha = 0.55$)	**41.5**	**23.5**	**36.3**

2-2 (Find the soft matches) needs to be repeated and typically the pattern length is not too large (technically, dynamic programming can be used here to achieve even better performance).

F EFFECTIVENESS OF SOFTMATCHA IN THE INFORMATION TASK

We conducted experiments to apply our method to information retrieval.

Soft-BM25 The well-known and strong baseline, BM25 (Robertson & Zaragoza, 2009), calculates relevance scores between a query and a document using term frequency (TF) and inverse document frequency (IDF), which involve counting occurrences of words or n-grams.

We implemented a soft version of BM25 (soft-BM25). For calculating the soft term frequency in a document, we compute the pattern-level score by multiplying the matching scores of each word in a query pattern, and then sum up for each pattern occurrence. Similarly, soft inverse document frequency is calculated by counting documents that softly contain the given query patterns. Then, we calculated the soft-BM25 score based on the most standard Lucene implementation.

To examine the effectiveness of soft matching on the information retrieval task, we compared the threshold parameter α between 1.0 and 0.55. $\alpha = 1.0$ means "not performing any semantic relaxation", making it equivalent to standard BM25. In contrast, $\alpha = 0.55$ allows for semantic relaxation of query pattern counting using our method, serving as a means to verify the effectiveness of the proposed method in information retrieval.

Benchmark dataset We used TREC-COVID dataset (Roberts et al., 2021), which is a part of Massive Text Embedding Benchmark (MTEB) (Muennighoff et al., 2023). This dataset consists of 171k documents and 50 queries (= instances). Notably, the original dataset comprised only queries formatted as natural language questions that are not suitable for queries of BM25 and soft-BM25; thus, we prepared the pattern queries for this experiment. Specifically, we manually transformed the natural language question, e.g., "how does the coronavirus respond to changes in the weather" to the queries for BM25 and soft-BM25, e.g., ["coronavirus", "response to weather", "weather change", "coronavirus response to weather changes"].

Evaluation metrics We evaluated the retrieval performance on the precision@20 (P@20), recall@1000 (R@1000), NDCG@20 following the previous work (Bendersky et al., 2020).

Results The experimental results are demonstrated in Table 8. The table shows that soft matching achieved better performance compared with exact matching in the information retrieval task.

To summarize this experiment, we confirmed that our `SoftMatcha` is effective not only in full-text search but also in the information retrieval task, which ranks relevant texts.