SoftMatcha: 大規模コーパス検索のための柔らかくも高速なパターンマッチャー

出口 祥之 [1*] 鴨田 豪 [2] 松下 祐介 [3] 田口 智大 [4] 末永 幸平 [3] 和賀 正樹 [3] 横井 祥 [5,2,6]
[1]NAIST [2] 東北大学 [3] 京都大学 [4] ノートルダム大学 [5] 国立国語研究所 [6] 理化学研究所
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

概要

用例や言語現象をコーパスから素早く探し出す grep などのパターン検索は，コーパスと人間をつなぐための基本的で重要な道具である．しかし，既存の文字列一致に基づくパターン検索では，表記揺れや類義語といった表層の変化を捉えることは難しい．文埋め込みを用いた密ベクトル検索も注目を集めており，意味的に粗く類似したテキストを検索できるが，具体的なクエリの出現位置の特定や列挙といった操作は困難である．本研究では，単語埋め込みを用いた柔らかいパターンマッチャー SoftMatcha を提案する．提案法は，転置索引を拡張したアルゴリズムを用いており，巨大コーパスに対して柔らかくも高速な検索・列挙をおこなうことができる．コーパス中の有害事例の列挙や，形態論的に複雑な特徴を持つ言語に対する用例検索を通して，SoftMatcha の有用性を実験的にも確認した*.

1 はじめに

自然言語処理（natural language processing; NLP）やコーパス言語学の近年の目覚ましい進歩は，巨大なコーパスの利活用に牽引されている．NLP 分野では巨大なコーパスを用いて学習された大規模言語モデル（large language model; LLM）がチャットボットや機械翻訳を実用レベルに引き上げ [1–4]，コーパス言語学では人間の言語の使用を統計的・数理的に明らかにするため膨大な言語資源がコンコーダンサ等を通じて参照されている [5, 6]．このような背景から，巨大コーパスに対する有用なパターン検索の需要はかつてないほど高まっている．例えば，大規模言語モデルによって有害情報・誤情報・プライバシー情報が生成されてしまった際には，その情報の発生源と思われる学習事例を特定するためにこうした技術が必要となる [7–9]．既存のパターン検索における課題は，マッチングの判定が文字列の表層表現に基づいている点にある []．自然言語は類義表現や言い換え表現を多分に含み，grep のような表層表現に基づく厳密な文字列マッチングでユーザーの要求を満たしづらい．一方で，文埋め込みを利用した密ベクトル検索 [10, 11] ではマッチングの柔軟性は上がるものの，文や文書全体に対する粗い意味的類似性によるマッチがおこなわれるため，クエリパターンの出現位置の列挙や検索は難しい．

表 1: 従来の文字列一致に基づくパターン検索（grep など）と単語埋め込みに基づく柔らかいパターン検索（提案法：SoftMatcha）による検索例.

クエリパターン	**Theorem 1**
従来のパターン検索	··· thanks to **Theorem 1** ···
柔らかい パターン検索 (SoftMatcha)	··· thanks to **Theorem 1** ··· ··· **Theorem 3** holds because ··· ··· By **Lemma 5**, we may assume ··· ··· **Equation 1** describes ···

本研究では，自然言語に内在する類義語や形態素の変化に対して柔軟に対応でき，かつ，10 億語規模の巨大なコーパスに対しても高速に検索できる，柔らかくも高速なパターンマッチャー SoftMatcha を開発した*．SoftMatcha の核となる柔らかいマッチングは，従来のパターンマッチにおける一致・不一致の二値，すなわち {0, 1} を，単語埋め込みによって連続値 [0, 1] へと連続化することにより実現される．また，従来の高速な文字列検索法である転置索引を，連続化されたマッチングに適用できるよう上手に拡張することで，SoftMatcha は 10 億規模のコーパスの中から柔らかくマッチする事例を 1 秒

* 現在，NTT コミュニケーション科学基礎研究所.
* ウェブツール: https://softmatcha.github.io/

未満で全列挙することができる．提案法の動作例を表 1 に示す．提案法は文字列一致に基づくパターン検索の結果を内包し，さらに表層表現が異なっていても意味的に類似した単語への置換を許容して検索できる．例えば，クエリに “Theorem 1” を与えたとき，“Lemma 5” といった事例を検索できる．

NLP やコーパス言語学における重要なタスク，すなわち有害事例の検出や，形態論的に複雑な特徴を持つ言語に対する用例検索の実験を行い，SoftMatcha の有用性を経験的にも確認した．

2 背景：テキスト検索

grep に代表される文字列検索により，クエリに与えた文字や単語の列（パターン）をテキストから探し出すことができる．テキスト上でのパターンの出現位置を取得でき，また，そのすべてを漏れなく列挙することができる．ただし，あくまで表層のマッチングであり，意味的に類似したパターンを探すことはできない．agrep [12] は編集距離に基づいて表層の揺れも許容させることでマッチングの柔軟性を上げた．しかし，編集距離も文字の一致・不一致に基づいており，表層の異なる類義語や表層の似た異義語などが頻出する自然言語には適さない．

密ベクトル検索 [10, 13–15] は，文埋め込みを用いてクエリとテキストの類似度が高い事例を取得する検索法である．表層が異なっていても意味的に類似したテキストまで探すことができるが，テキスト全体に対する粗い類似性に基づくため，具体的な用例の出現位置や出現頻度を求めることはできない．

3 提案法：SoftMatcha

提案する SoftMatcha は，文字列一致に基づくパターンマッチャーを，単語埋め込みの類似度を用いて連続化することにより，自然言語特有の表記揺れや類義語も含む柔軟な検索を可能とする．また，高速な文字列検索法の転置索引を連続化することで，柔らかい状態でかつ高速な検索を実現する．

SoftMatcha のウェブツールを一般公開した†．ぜひ試していただきたい．以下，図 1 を用いながら，具体的なアルゴリズムを示す．実際にできることに特に興味がある読者は 4 節に飛んで差し支えない．

柔らかいマッチ　テキストを $\boldsymbol{t} \coloneqq (t_1, \cdots, t_N) \in \mathcal{V}^*$，クエリパターンを $\boldsymbol{p} \coloneqq (p_1, \cdots, p_n) \in \mathcal{V}^*$ とする．$\mathcal{V}$ は単語集合．パターン検索とは，テキスト中に含まれるパターンの開始位置の集合 $\left\{ i \in \{1, \ldots, N\} \,\middle|\, \forall k \in \{1, \ldots, n\}.\, p_k = t_{i+k-1} \right\}$ を求める操作である．ここで，**単語対 $(v, v') \in \mathcal{V}^2$ が柔らかくマッチする**という二項関係，つまり単語埋め込み間のコサイン類似度が α 以上であることを表す $v \approx_\alpha v' \overset{\triangle}{\iff} \cos(E(v), E(v')) \geq \alpha$ を導入する．なお，$E \colon \mathcal{V} \to \mathbb{R}^D$ は単語を埋め込み表現に変換する関数，$D \in \mathbb{N}$ は単語埋め込みの次元数，$\alpha \in (0, 1]$ は閾値を表す．

索引化　提案法は，まず事前にテキストから単語単位の転置索引 I を構築する（図 1a）．すなわち，語彙の各単語 $v \in \mathcal{V}$ がテキスト上で出現した位置 $I_v \coloneqq \left\{ i \in \{1, \ldots, N\} \,\middle|\, v = t_i \right\}$ を記録する．SoftMatcha で用いる転置索引のデータ構造や構築時の処理は通常の文字列検索で用いられるものと同じであり，単語 v の出現位置をテキストから探索することなく高速に列挙することができる []．

柔らかいパターン検索　以上を用いて，検索結果 $M \coloneqq \left\{ i \in \{1, \ldots, N\} \,\middle|\, \forall k \in \{1, \ldots, n\}.\, p_k \approx_\alpha t_{i+k-1} \right\}$ を求めれば柔らかいパターン検索が実現する．

1. パターン $\boldsymbol{p}$ 中の各単語 p_k について，語彙に対して柔らかくマッチする単語の集合 $S_k \coloneqq \left\{ v \in \mathcal{V} \,\middle|\, v \approx_\alpha p_k \right\}$ を求める．
2. 転置索引を用いて柔らかくマッチするパターンの開始位置を求める：
   2-1. 転置索引 I から，S_k 中の各単語に対応する索引の結合 $\tilde{I}_k = \bigcup_{v \in S_k} I_v$ を得る．
   2-2. $\tilde{I}_k$ の各要素の値を $k-1$ 引く操作を $\tilde{I}_1, \cdots, \tilde{I}_n$ にそれぞれ適用した後，それらの共通部分を求めて出力する．すなわち，$M = \bigcap_{k=1}^{n} \left\{ i - (k-1) \,\middle|\, i \in \tilde{I}_k \right\}$ を出力する．

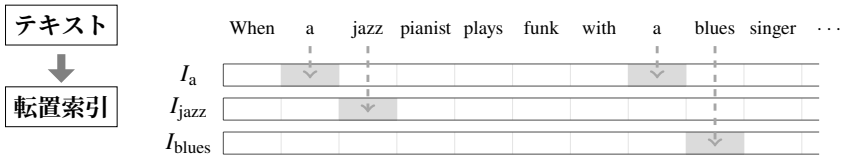
4 実験

4.1 大規模コーパス中の有害事例検索

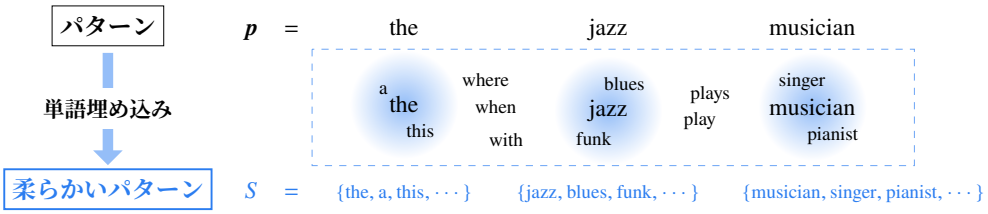
NLP への応用例として有害テキストのフィルタリングを想定し，10 億語規模の巨大なコーパスから有害事例を検索する実験をおこなう．

設定　検索対象のテキストとして，LLM-jp v2.0 コーパス [16] の中の英語 Wikipedia（34 億単語）と日本語 Wikipedia（11 億単語）のデータを使用した．英語 Wikipedia では “*homemade bombs*” を検索し，日本語 Wikipedia では “手製爆弾” を検索した．単語埋め込みと閾値 α として，英語は GloVe

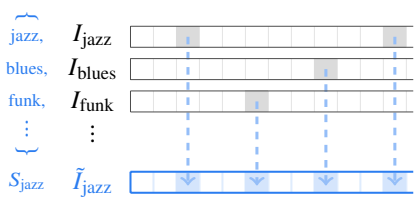
† https://softmatcha.github.io/



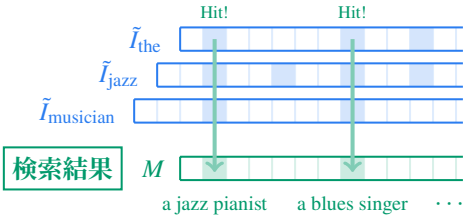
(a) 前処理：テキストから転置索引を構築.



(b) 検索手順 1: 単語埋め込みを用いて柔らかくマッチするパターンを求める.



(c) 検索手順 2-1: 柔らかくマッチする単語の索引を結合.



(d) 検索手順 2-2: パターンの開始位置を求める.

図 1: 提案法の柔らかいパターン検索の概要図.

の glove-wiki-gigaword-300 [2] と $\alpha = 0.55$ にを，日本語は fastText の facebook/fasttext-ja-vectors [17] と $\alpha = 0.50$ を用いた．ベースラインとして，文字列一致に基づく転置索引を用いた全文検索（文字列検索），および，文埋め込みモデルの intfloat/multilingual-e5-large [11] を用いた密ベクトル検索を採用した‡．文字列検索と SoftMatcha には 152 コアの CPU（Intel® Xeon® Platinum 8368 CPU @ 2.40GHz）と 226 GiB の主記憶を搭載した計算機を用い，密ベクトル検索のためのテキストの埋め込みには NVIDIA A100 の GPU を 8 基用いた．

検索結果　表 2 に，10 億単語規模コーパスの検索結果を示す．密ベクトル検索ではクエリの最近傍 10 件を探索しうち数件を示してある．表に示すとおり，我々の SoftMatcha は，文字列検索よりも柔軟に意味的に類義するパターンを検索できている．なお SoftMatcha の検索結果には文字列検索の結果が内包されてるため，SoftMatcha は文字列一致に基づくパターン検索を拡張した道具だといえる．また，密ベクトル検索の結果を見ると，たとえば日本語ではクエリとの関連性の低い “花火” の記事など，関連性の低いパターンもヒットしており，単語や句レベルの用例の検索に適さないことがわかる．また，SoftMatcha はクエリパターンやクエリに類似するパターンが現れる箇所を文字列検索と同様に列挙できることにも注目されたい．密ベクトル検索は，文字列検索や SoftMatcha と異なり，クエリパターンがテキスト中のどこに位置しているのかの特定・列挙が難しい．

実行速度　表 3 に，コーパスの索引化と検索にかかった実行時間を示す．まず索引化について，SoftMatcha の転置索引の構築は通常のもの同様の形式をとっており，つまり文字列検索と同じ計算量で索引化することができる．検索について，文字列検索が最も速く，SoftMatcha はその次に速く，密ベクトル検索が最も遅いという結果になった．SoftMatcha は文字列検索の結果を内包し，検索結果が多くなる分，文字列検索よりは遅くなったと考えられる．密ベクトル検索は，検索時にニューラルモデルを用いてクエリを埋め込む処理が含まれるため，最も遅い．まとめると，SoftMatcha は通常の転置索引を用いた文字列検索よりはやや遅いものの，34 億単語の英語 Wikipedia コーパスを 0.1 秒以内と実用的な超高速度で検索できることができる．

‡ LLM の訓練に実際に使用された訓練事例ごとにテキストを埋め込み，Faiss [18] の実装により，HNSW [19] を用いたグラフに基づく近似近傍探索を行った．

表 2: 英語・日本語の Wikipedia コーパスに対する検索結果.

	文字列検索	SoftMatcha	密ベクトル検索
クエリ：“*homemade bombs*”（英語 Wikipedia）			
ヒット数	107	1,473	n/a （取得件数 k に依存）
ヒット例	*homemade bombs*	*homemade bombs*	記事：Survival Under Atomic Attack
		home-made grenades	記事：Mark 24 nuclear bomb
		homemade missiles	記事：List of common misconceptions
クエリ：“手製爆弾”（日本語 Wikipedia）			
ヒット数	27	42	n/a （取得件数 k に依存）
ヒット例	手製爆弾	手製爆弾	記事：花火
		手製手榴弾	記事：手持ち花火

表 3: 英語・日本語それぞれの Wikipedia コーパスの索引化と検索の実行時間（秒）.

	英語		日本語	
	索引化	検索	索引化	検索
文字列検索	685.8	0.005	242.5	0.022
SoftMatcha	685.8	0.098	242.5	0.055
密ベクトル検索	1036.5	0.389	320.4	0.283

4.2 形態論的に複雑な言語の用例検索

コーパス言語学への応用可能性を検証するため，形態論的に複雑な言語であるラテン語のコーパスから用例を検索する実験を行った.

設定 検索対象のコーパスには，500 万単語からなる Perseus Project [20] コーパスと，10 万単語からなる Augustinian Sermon Parallelism (ASP) データセット [21] を用いた．単語埋め込みには fastText の `facebook/fasttext-la-vectors` [17] を用いた．クエリパターンとして，“*factus est*”（彼/それはなされる/作られる）を検索した.

検索結果 表 4 はクエリ *factus est*（それ/彼はなされる/作られる）に対するマッチをまとめたものである．明らかに SoftMatcha はクエリを意味論的に近い単語たちへとにマッチさせている．例えば，クエリ *factus*（なされる/完了される/作られる）には *mortuus*（死んだ）と *creatus*（創られた）がマッチした．興味深いことに，このツールは語彙素が同じであるが形態論的特徴が異なるような語形の違いを捉えられている（ピンクで強調した）．例えば，クエリ中のラテン語のコピュラ動詞 *est*（である）は非常に不規則な活用パターンを示すが，このツールでは正しく *sunt*，*esset*，*erat* といった活用形がマッチした．さらに，クエリ語 *factus* に対する *mortuus* や *creatus* といったマッチは，意味論的に近いだけでなく，形態論的特徴（完了，分詞，男性，主格，単数）も共通している.

表 4: SoftMatcha によるラテン語検索の例．*factus est*（それ/彼はなされる/終えられる/作られる）行間のグロスのうち，1 行目は単語を，2 行目は形態論的分析を，3 行目は意訳を表す．形態論的マッチはピンクで，意味論的マッチは青で強調した．括弧内の数値はクエリ語と対応するマッチ語との埋め込み空間でのコサイン類似度を表す.

Query: *factus est*
fact-us *est*
do.PASS.PF.PTCP-M.NOM.SG be.IND.PRS.3SG
‘he/it is done/finished/made’
Match: *facta sunt* (0.56, 0.56)
fact-a *sunt*
do.PASS.PF.PTCP-N.NOM.PL be.IND.PRS.3PL
‘they are done’ or ‘they are facts’
Match: *mortuus esset* (0.53, 0.58)
mortu-us *esset*
die.ACT.PF.PTCP-M.NOM.SG be.SUB.IMPF.3SG
‘he/it was dead’
Match: *creatus erat* (0.65, 0.65)
creat-us *erat*
create.PASS.PF.PTCP-M.NOM.SG be.IND.IMPF.3SG
‘he/it was created’

5 おわりに

単語埋め込みの類似度に基づく柔らかいマッチングにより柔軟なパターン検索を実現する SoftMatcha のアルゴリズムを提案し，ウェブツールを公開した．提案法は，従来の転置索引を連続的に拡張し，10 億単語規模のコーパスに対しても高速にパターン検索することができる．今後は，NLP やコーパス言語学における検索を利用したさまざまなタスクに対してのさらなる応用可能性を検討していきたい.

謝辞

本研究の実験は，LLM-jp（https://llm-jp.nii.ac.jp）の言語資源と計算資源を利用して行われました．また本研究は，JSPS 科研費 22H05106, 24KJ0133, および JST ACT-X JPMJAX200U, JST ACT-X JPMJAX200S, JST さきがけ JPMJPR22CA, JST CREST JPMJCR2012 の支援を受けたものです．ここに感謝の意を表します．

参考文献

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. **arXiv [cs.CL]**, 16 January 2013.

[2] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)**, pp. 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[3] 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, 2019. Association for Computational Linguistics.

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

[5] 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.

[6] Frank Smadja. Retrieving Collocations from Text: Xtract, 1993.

[7] Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. A Survey on Automated Fact-Checking. **Transactions of the Association for Computational Linguistics**, Vol. 10, pp. 178–206, 2022.

[8] 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, 2024.

[9] 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, editors, **Proceedings of the 16th International Natural Language Generation Conference**, pp. 28–53, Prague, Czechia, September 2023. Association for Computational Linguistics.

[10] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wentau 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.

[11] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual e5 text embeddings: A technical report, 2024.

[12] Sun Wu and Udi Manber. Fast Text Searching Allowing Errors. **Commun. ACM**, Vol. 35, No. 10, pp. 83–91, 1992.

[13] 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.

[14] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised Dense Information Retrieval with Contrastive Learning, 2022.

[15] 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, 2024.

[16] LLMjp: Akiko Aizawa, Eiji Aramaki, Bowen Chen, Fei Cheng, Hiroyuki Deguchi, et al. LLM-jp: A Cross-organizational Project for the Research and Development of Fully Open Japanese LLMs, 2024.

[17] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning Word Vectors for 157 Languages, 2018.

[18] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library, 2024.

[19] 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.**, Vol. 42, No. 4, p. 824–836, April 2020.

[20] Gregory Crane. The Perseus Digital Library and the future of libraries. **International Journal on Digital Libraries**, 2023.

[21] 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, editors, **Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing**, pp. 5007–5039, Singapore, December 2023. Association for Computational Linguistics.

A　付録

図 2: SoftMatcha を用いた検索例.

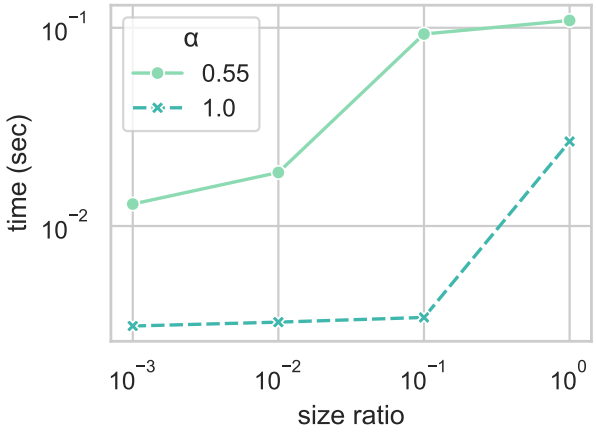


図 3: テキストをサンプリングして英語 Wikipedia コーパスのサイズを変化させたときの検索速度.

図 2 に，SoftMatcha の検索例を示す．また，図 3 に，検索対象のコーパスサイズを変えたときの SoftMatcha の検索時間を示す.